Sonderheft Nr. 60
Preis DM 18.50
öS 140.–, sfr. 18.50
ELO

Bit für Bit zum Selbstbaucomputer

Funktionsweise und Schaltungsdetails, dargestellt am Eigenbau-Computer MOPPEL



Franzis'

Die ELO – der lockere Einstieg in die Elektronik und Mikrocomputerei.

Ob Sie sich ein faszinierendes Hobby erschließen möchten oder einfach nur Bescheid wissen wollen: In ELO steht drin, was Spaß macht oder Wissen schafft.

ELO zeigt, wie moderne Technik funktioniert. Mikrocomputertechnik, Audio und Video, Musikelektronik, Meßtechnik, die neuen Medien oder raffinierte Modellbau-Elektronik.

Und das ganz Besondere: ELO bringt jede Menge Bauanleitungen. Da können Sie ganz praktisch in die faszinierende Technik einsteigen, die unsere Welt verändert, über die alle reden und von der bishe. leider die wenigsten Leute viel verstehen.

Mit ELO-Hilfe werden Sie beginnen, selbst elektronische Geräte zu bauen. Erst einfache – und wenn Sie später wollen, auch kompliziertere

Mit ELO steigen Sie ein. Ganz locker. Probieren Sie es aus. Die Kennenlern-Karte an der Rückseite dieses Heftes macht es leicht und risikolos. Vielleicht erschließt sie gerade Ihnen mehr als nur ein faszinierendes Hobby...





Magazin für Elektronik und Mikrocomputerei

Moppeln Sie mal



MOPPEL ist das modulare Prozessor-Programm der ELO, ein Selbstbau-Computer, dessen Bau- und Funktionsbeschreibung Sie seit Jahren in der ELO-Mikrocomputer-Rubrik verfolgt haben. Das vor Ihnen liegende Heft faßt alle Beiträge zusammen, die bisher zu diesem Komplex erschienen sind; wir erfüllen damit einen häufig geäußerten Wunsch all derer, die diese Artikel nicht lückenlos gesammelt haben.

Was verbirgt sich nun hinter dieser Konzeption, die den ohnehin überschwemmten Markt um eine weitere Variante bereichert? Ziel bei der Entwicklung dieses Prozessor-Systems war es, ein vielseitiges, modulares und jederzeit erweiterbares Angebot zu schaffen, das auf die Belange des Hobbyisten zugeschnitten ist. Dem Anfänger bietet sich damit die Möglichkeit, sich in Maschinensprache in die Mikrocomputer-Technik einzuarbeiten und anschließend sein System Schritt für Schritt zu erweitern, so wie es sein Wissensstand und die finanziellen Möglichkeiten zulassen. Auf diese Weise entsteht ein voll ausgebautes Entwicklungssystem, das sich den unterschiedlichen Erfordernissen und Schwerpunkten anpaßt; so ist es beispielsweise möglich, mit dem ROM-residenten BASIC-Interpreter die allseits publizierten Programmbeispiele nachzuvollziehen, oder man kann in einer anderen Konfiguration natürlich auch das Gewicht auf ein ausgebautes Computer-System legen, in dem EPROM-Programmierzusatz, Assembler und diverse externe Schnittstellen die optimale Entwicklung eigener Programme unterstützen.

MOPPEL ist in jeder Hinsicht maschinennah konzipiert worden, um die oftmals anzutreffende praxisfremde Sterilität anderer Systeme zu vermeiden. Was nützen die ausgefeiltesten Programmtechniken, wenn man nicht in der Lage ist, mit einem Bit eine Leuchtdiode oder ein Relais zu aktivieren? Und was ist ein BASIC-Interpreter wert, der nicht wie selbstverständlich auch in der Maschinen-Ebene zu Hause ist? Beim Umgang mit dem Computer sollte der Bezug zum Bit nie verlorengehen, damit zu keinem Zeitpunkt das unheimliche Gefühl aufkommt, die Maschine vor einem könnte auf dumme Gedanken kommen und ihren Meister in die Pfanne hauen. In einem Computer wird „auch nur mit Wasser gekocht", d.h. mit Standard-ICs und TTL-Pegeln herumgewirtschaftet. Und wer wie hier die Möglichkeit hat, sich sein System sogar selbst zusammenzulöten, der hat mit Sicherheit ein anderes (und vor allem gesundes) Verhältnis zur Computerei als die neunmalklugen Zeitgenossen, die sich mit Randspalten-Bildung und Schlagwörtern begnügen.

Sehen Sie das hier vorgestellte System darum nicht in Konkurrenz zu anderen, sehr spezifischen Geräten, die ungleich billiger herstellbar sind. Hochwertiges Material und die modulare Konzeption haben ihren Preis, der sich in der Praxis aber schnell bezahlt macht (s.o.). Und hinzu kommt eine (nicht zu Unrecht) stolz geschwellte Brust, wenn sich der selbst zusammengebaute Computer endlich regt und den Dialog mit seinem Meister aufnimmt.
Wenn Sie den Einstieg auf unterster Ebene suchen und sich zunächst mit HEX-Tastatur und Siebensegment-Anzeige begnügen, nehmen Sie sich das Heft von Anfang an vor. Sollten Sie dagegen zielstrebig das bildschirmorientierte System mit Schreibmaschinen-Tastatur und Video-Interface ansteuern, können Sie die einleitenden Abschnitte getrost überschlagen. Natürlich bieten die detaillierten Grundlagen-Artikel auch demjenigen einen reichen Fundus, der nicht zum Lötkolben greifen will, sondern der sich nur über die Arbeitsweise der verschiedenen Baugruppen informieren will. Und seien Sie ehrlich - selbst als „alter Hase" der Mikrocomputer-Technik bleiben viele Zusammenhänge im Dunkeln, denkt man nur an den Anschluß peripherer Baugruppen mit der zugehörigen Decodier-Logik, die aus dem Spaghetti der ineinander verschachtelten Signale geeignete Steuerimpulse formt. Und unter diesem Aspekt wünschen wir Ihnen eine vergnügliche Lektüre dieses Heftes, und wenn Sie in die Hardware einsteigen wollen, viel Spaß bei der Arbeit. Trauen Sie sich getrost auch den Selbstbau Ihres Computers zu, was bei einigermaßen Löterfahrung überhaupt keine Affäre ist. Und in diesem Sinne wünsche ich Ihnen ein paar kurzweilige Stunden, wenn Sie daran gehen, selbst ein bißchen zu „moppeln"

Ihr Reinhard Gößler

(Reinhard Gößler)

Inhalt

1984
Franzis-Verlag GmbH, Karlstraße 37–41, 8000 München 2
Sonderheft der Zeitschrift ELO
Redaktion: Reinhard Gößler, für den Text verantwortlich: Brigitte Kriebel
Herstellung und Layout: Günter Ropertz

ISSN 0172-2786
Druck: Franzis-Druck GmbH, Karlstraße 35, 8000 München 2
ZV-Artikel-Nr. 60031 – F/ZV/184/897/10'

Mein Name ist MOPPEL

Wir beginnen hier die Vorstellung von MOPPEL, dem modularen Prozessor-Programm der ELO. Es basiert auf dem Mikroprozessor 8085A, den Sie vielleicht von unserem Mikrocomputer UMS-85 bereits kennen. MOPPEL spricht aber nicht nur dieselbe Sprache, sondern kann selbstverständlich auch alle Interface-Karten ansteuern, die wir im Zusammenhang mit dem UMS-85 beschrieben haben. Zu der Neuentwicklung haben wir uns entschlossen, um Ihnen die Möglichkeit zu geben, sich schrittweise ein voll ausgebautes Mikrocomputer-System zusammenzustellen, das über eine Schreibmaschinen-Tastatur und Bildschirm verfügt und sich später auch in BASIC programmieren läßt (**Tabelle 1**).
Das System arbeitet prinzipiell mit einfacher 5-V-Versorgung; es verwendet Europakarten (160×100 mm^2) mit 64poligen VG-Steckverbindern, die zwar nicht gerade billig sind, aber die Gewähr für ein Höchstmaß an Funktionssicherheit bieten. Ein Blick auf Tabelle 1 zeigt Ihnen, welche Baugruppen in diesem Rahmen verfügbar sind.
Um ein Anwender-System individuell und flexibel zusammenstellen zu können, werden die Karten an einem gemeinsamen Bus betrieben (**Tabelle 2**; weitgehend ECB-Bus). Sämtliche Karten sind als Bausatz oder fertig bestückt und 100 % getestet erhältlich. Sie werden jeweils in einer Basisversion angeboten, zu der eine Erweiterung verfügbar ist; die Basisversion stellt die preiswerte, aber bereits voll funktionsfähige Grundausbaustufe dar, und die Erweiterung nutzt weitergehende Feinheiten aus (bei der CPU gehören hierzu u. a. der Puffer-Akku und die 25polige Buchsenleiste für die serielle Schnittstelle).

Tabelle 1.

Mikrocomputer mit RAM/EPROM
HEX-Tastatur/Anzeige
Miniatur-Netzteil
Cassetten-Interface
EPROM-Programmierzusatz
5-V-Netzteil
ASCII-Tastatur
Tastatur-Erweiterung
Video-Interface
Echtzeit-Uhr
RAM-/ROM-Karte
Bus-Platine
Parallel-Interface
Serielles Interface, Timer
Thermodrucker
Einzelschritt-Modul

Im HEX-Code geht es zur Sache

In der Minimalkonfiguration wird der Computer in Maschinensprache programmiert, genauso, wie Sie es von unseren bisherigen Programmbeispielen her kennen. Dazu läßt sich die HEX-Tastatur- und Anzeige-Platine des UMS-85 anschließen, und man kann hierüber u. a. Daten eingeben, Register inspizieren und natürlich auch Anwenderprogramme ausführen. Zum Betrieb ist eine einfache 5-V-Versorgung (Belastung mit 500 mA) erforderlich, und passend dazu wird das Miniatur-Netzteil angeboten (**Bild 1**). Es stellt außer der 5-V-Systemversorgung noch ±12 V zur Verfügung, um die serielle Schnittstelle auf der Zentraleinheit (falls gewünscht) betreiben zu können.
Ein Blick auf das Blockschaltbild der Zentraleinheit (**Bild 2**) läßt bereits die Systemkonzeption erkennen: der Mikroprozessor 8085A mit Quarzoszillator und obligatem Adreß-Latch 74LS373, die verschiedenen Treiber-ICs (74LS244, '245, '04) stellen die erforderliche Leistung bereit, um die übrigen am Bus „hängenden" Karten betreiben zu können. Auch den Anschluß der Tastatur über eine IC-Fassung mit Flachbandkabel ist Ihnen nicht neu, denn beim UMS-85 geschah das ganz genauso; auf der MOPPEL-Zentraleinheit ist allerdings eine 20polige Fassung vorgesehen, in die das 16polige Flachbandkabel der HEX-Tastatur linksbündig einzustecken ist (die restlichen Pins sind zur Ansteuerung der ASCII-Tastatur vorgesehen).

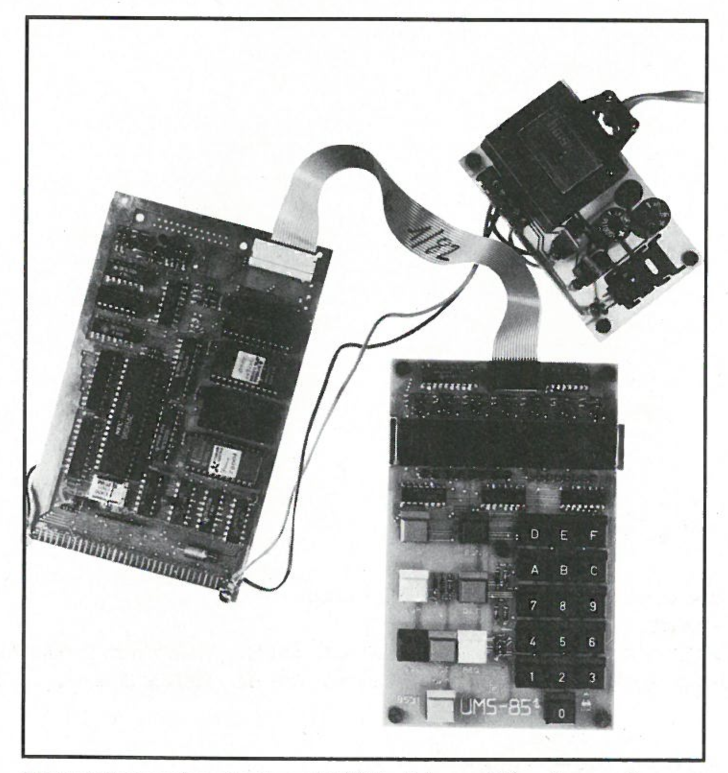


Bild 1: Mit dem Anschluß von HEX-Tastatur und Anzeige sowie einem Netzteil entsteht ein betriebsfertiger Mikrocomputer.

Ganz nach Wunsch: Die Speicherbestückung

Die Adreßdecodierung sorgt (wie in jedem Computer-System) für die Zuordnung von Software-Adresse und Hardware-Speicherblock. Auf der MOPPEL-CPU sind vier Steckplätze für Speicher vorgesehen (Platz #0...#3); diese können je nach Systemausbau und Anwendungsfall bestückt werden, wobei die Schaltung der Brücken 1...4 zu beachten ist (vgl. auch Bild 2): Die Plätze 0 und 1 sind immer gleich zu bestücken; beim Einsatz von 2716-EPROMs (2 KBytes) ist die Brücke 1 einzusetzen, und wenn Sie die doppelt so großen Typen 2732 einsetzen (4 KBytes), muß die Brücke 2 geschlossen werden. Auf Platz 3 können Sie entweder ein RAM oder ein EPROM einsetzen, das jeweils 2 KBytes Kapazität besitzt (2716 oder 6116-CMOS-RAM); im ersten Fall setzen Sie Brücke 3 ein, und beim RAM ist Brücke 4 erforderlich. Auf Platz 3 wird ein CMOS-RAM mit 2 KBytes Kapazität plaziert (6116); es kann mit einem Puffer-Akku versorgt werden, um auch bei abgeschalteter Versorgungsspannung die Daten zu halten. Wie Sie vermutlich wissen, hat der 8085A einen 16 Bit breiten Adreßbus; damit kann er einen Speicherbereich von 2^{16} = 65 536 Adressen ansteuern; da dieser riesig scheinende Adreßraum im Ernstfall doch zu klein werden könnte, wurden beim MOPPEL drei zusätzliche Adreßbits A16...A18 generiert. Um zu zeigen, daß dies quasi „künstliche" Adreßbits sind (die der Hersteller nicht vorgesehen hat), tragen sie das kleine Sternchen (vgl. Tabelle 2). MOPPEL kann folg-

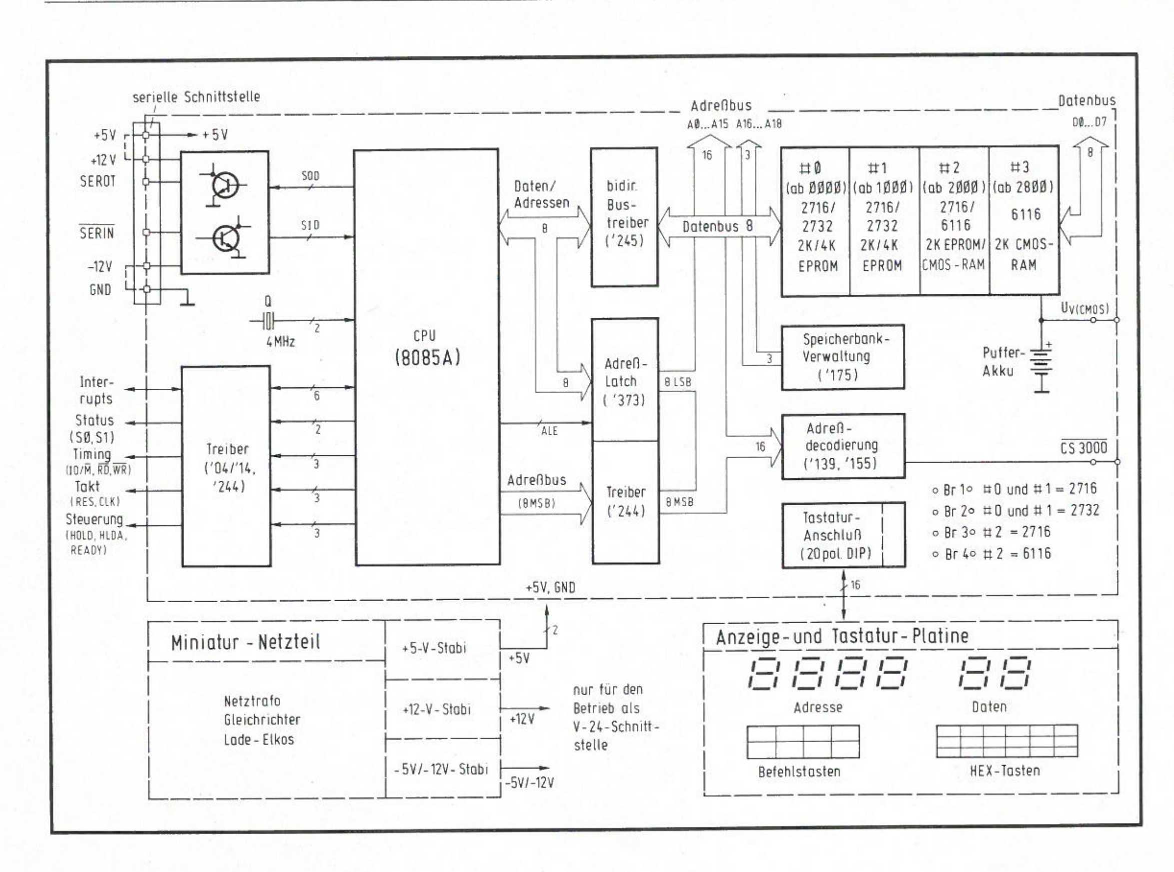


Bild 2: Blockschaltung der Zentraleinheit.

lich einen Adreßraum von 8×64 KBytes ansprechen; auf der Zentraleinheit sind maximal 10 KBytes EPROM und 4 KBytes RAM (zusammen maximal 12 KBytes) möglich. Das dürfte auch für unersättliche Programmierer erst einmal ausreichen, um auch an längsten Winterabenden ausreichend Beschäftigung zu haben!

Der direkte Ein- und Ausgang

Am 8085A sind je ein 1-Bit-Ein- und -Ausgang vorhanden, um beispielsweise den seriellen (ein Bit nach dem anderen) Datenverkehr zu ermöglichen; diese beiden Leitungen sind auf der MOPPEL-CPU über Transistorstufen gepuffert worden und bilden eine serielle Schnittstelle des Systems; hier kann beispielsweise das langsame Cassetten-Interface zur externen Speicherung von Programmen und Daten angeschlossen werden. Es ist aber auch möglich, an diesem Anschluß komplette Bediengeräte mit Bildschirm und Tastatur (Terminals) zu betreiben. Nur in diesem Fall müssen extern noch ±12 V angeschlossen werden, weil dies die entsprechende Norm vorschreibt. Für den normalen Betrieb jedoch entfällt dieser Zusatz, und das Cassetten-Interface kommt selbstverständlich mit den systemeigenen 5 V aus (es verbindet den für +12 V vorgesehenen Anschluß einfach mit +5 V und den −12-V-Anschluß legt es an Masse). In dieser Konfiguration können Sie Ihren Computer in Maschinensprache programmieren, wobei ein im 2716-EPROM abgespeichertes Monitor-Programm die entprechende Verwaltung übernimmt. Um sicherzugehen, daß im Abschaltaugenblick die im RAM enthaltenen Daten erhalten bleiben, sollte man die RESET-Taste drücken. Andernfalls kann diejenige RAM-Zelle ungewollt modifiziert werden, auf die das System gerade zugreift; im Monitor-Betrieb ist das die unterste RAM-Zelle 2800. (Für die Pufferung muß auf der CPU der Akku bestückt sein.)

Die Speicherbestückung

Das beginnt mit dem Detailschaltplan (**Bild 3**), dessen funktionellen Aufbau Sie bereits anhand des Blockschaltbildes (Bild 2) kennengelernt haben. Als wesentliche und ergänzende Feinheit sei hier die Speicherbestückung und der daraus resultierende Einsatz der Brücken Br. 1...4 erwähnt; Sie sehen noch einmal, daß die Speicher auf den Plätzen 0 und 1 gleich bestückt werden müssen (2-K- bzw. 4-K-EPROM und Brücke 1 bzw. 2). Mit Brücke 3 bzw. 4 wird festgelegt, ob auf Platz 2 ein 2-K-EPROM (2716) oder ein pinkompatibles CMOS-RAM (6116) eingesetzt wird. Mit dem Akku allerdings wird nur das RAM auf Platz 3 gepuffert. Dazu ist es entscheidend wichtig, als CS-Treiber (IC 11) ein Standard-TTL-IC vom Typ 7432 einzusetzen; die LS-Version wird im stromlosen Zustand niederohmig und „lutscht" den Akku binnen kurzer Zeit leer.

Mit dem Lötkolben ans Werk

Bei der Bestückung nach **Bild 4** sind nur ein paar Feinheiten zu beachten. Dazu gehört beispielsweise die Polung der Elkos, deren Einsetzen Ihnen durch ein kleines „+" auf der Platine erleichtert wird. Auch die Ausrichtung der Dioden wird durch einen kleinen Katodenpfeil auf der Karte gekennzeichnet, aber Sie achten bitte auf die unterschiedlichen Si- bzw. Ge-Typen (Ge-Typen wegen der geringeren Durchlaßspannung in der CMOS-

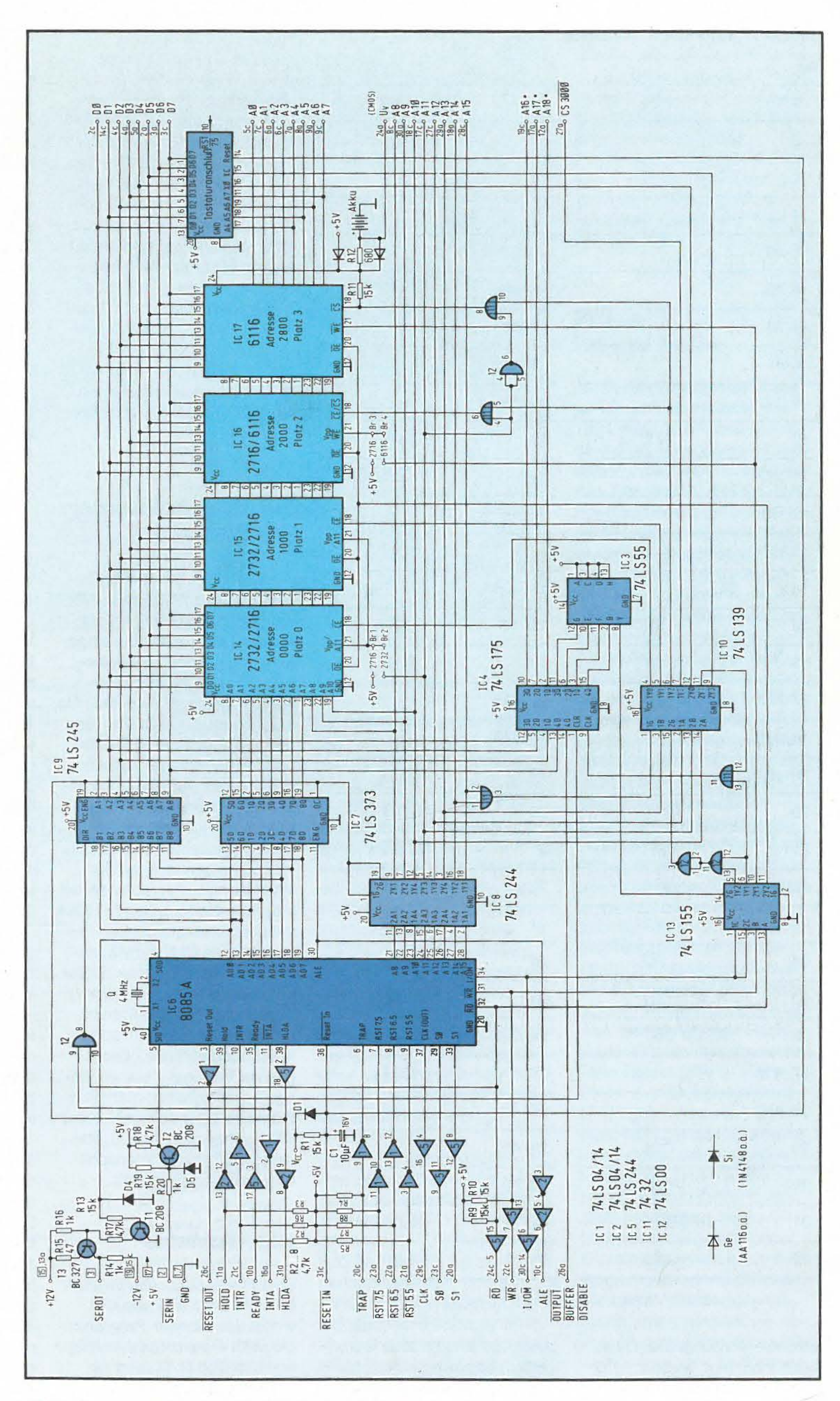


Bild 3: Detailschaltung der MOPPEL-Zentraleinheit.

Tabelle 2. MOPPEL-Busbegleitung.

c		a	
1	+5 V	1	+5 V
2	D0	2	D5
3	D7	3	D6
4	D2	4	D3
5	A0	5	D4
6	A3	6	A2
7	A1	7	A4
8	A8	8	A5
9	A7	9	A6
10	ALE	10	READY
11	VID	11	$\overline{\text{HOLD}}$
12	HS	12	∗A18
13	VS	13	+12 V
14	D1	14	BAS
15	−12 V	15	−5 V
16	PRTOT	16	INTA
17	A11	17	∗A17
18	A10	18	A14
19	∗A16	19	PRTIN
20	$\overline{\text{TRAP}}$	20	S1
21	$\overline{\text{INTR}}$	21	$\overline{\text{RST 5.5}}$
22	$\overline{\text{WR}}$	22	$\overline{\text{RST 6.5}}$
23	S0	23	$\overline{\text{RST 7.5}}$
24	$\overline{\text{RD}}$	24	$U_{V(CMOS)}$
25	CIN	25	$\overline{\text{CS 3000}}$
26	$\overline{\text{RESET OUT}}$	26	$\overline{\text{Outp. Buff. Disable}}$
27	A12	27	COT
28	A15	28	
29	CLK	29	A13
30	IO/$\overline{\text{M}}$	30	A9
31	$\overline{\text{RESET IN}}$	31	$\overline{\text{HLDA}}$
32	GND	32	GND

Stromversorgung). Der Quarz muß unbedingt gegen die darunterliegenden Leiterbahnen isoliert werden (Papierscheibchen); ein am Gehäuse festgelöteter Haltebügel schafft für das Gehäuse definierte Pegelverhältnisse. Quarze, die ohne Halt in der Luft herumbammeln, sind nicht nur unästhetisch, sondern können sich unter Umständen auch Störspitzen einfangen. Das Widerstandsnetzwerk muß mit dem Punkt zum kleinen „+" zeigen (gemeinsamer Anschluß der sieben internen Widerstände). Zum einfachen Anschluß der Stromversorgung sind im Bausatz zwei Lötösen vorgesehen, die mit den M2,5-Schrauben an der Plus- bzw. Masse-Sammelleitung befestigt werden. Da ansonsten alle ICs gleich ausgerichtet sind, sollte es für den geübten Löter ein leichtes sein, den Aufbau herzustellen.

Ein alter Bekannter: die Tastatur

Die Verbindung zwischen Tastatur und Zentraleinheit stellt ein Flachbandkabel her, das auf der Tastatur-Platine eingelötet und auf der CPU **linksbündig** in die 20polige Fassung eingesteckt wird (die z. Zt. freien vier Pins sind für den späteren Anschluß einer Schreibmaschinen-Tastatur vorgesehen). Auf die in **Tabelle 2** angegebene MOPPEL-Busbelegung kommen wir noch im einzelnen zurück (u. a. im Zusammenhang mit der Vorstellung der verschiedenen Interface-Karten). Das EPROM mit dem internen Monitor-Programm, das CMOS-RAM sowie der Mikroprozessor 8085A kommen auf die mitgelieferten Fassungen, die TTL-ICs sollten eingelötet werden, sofern keine hochwertigen Fassungen zur Verfügung stehen (billige Typen schaffen mehr Probleme als irgendeinen Nutzen; TTLs gehen nicht kaputt, solange man nicht unmenschliche Gewalt anwendet!).

HEX-Tastatur

Die LED-Anzeige wird im Multiplex-Betrieb angesteuert, wobei das Monitor-Programm die MUX-Rate softwaremäßig erzeugt (**Bild 5**). Dies ist ein Schaltungskonzept mit extrem geringem Hardware-Aufwand, bei dem jedes einzelne Segment der Anzeige per Programm ein- und ausgeschaltet werden kann.
Auch die Tastatur ist in Form einer Matrix aufgebaut, um sie rationell nach Zeilen und Spalten abfragen zu können. Bei der Bestückung ist darauf zu achten, daß die Digit-Treiber-Transistoren (unterhalb der Anzeigen) PNP-Typen sind (BC327 o. ä.), und die oberhalb der Anzeigen liegenden Segmenttreiber sind NPN-Transistoren (BC208 o. ä.; **Bild 6**). Bei den LEDs zeigt die Katode (das kurze Anschlußbein bzw. die abgeflachte Seite) immer zum unteren Platinenrand. Nach dem Einlöten der Anzeigen folgt die Montage der roten Filterscheibe, die beidseitig von einem Drahtbügel gehalten wird.
Wenn Sie nun beide Platinen über das Flachbandkabel miteinander verbinden (auch hier ist Vorsicht geboten, die zarten Stifte an den IC-Steckern können leicht verbiegen oder brechen!), muß sich der Monitor melden (wenn das Netzteil, wie folgt, bereits angeschlossen ist): In der Anzeige erscheint linksbündig die Adresse 2800 (die unterste im Arbeitsspeicher), und rechts wird der Inhalt dieser Speicherstelle dargestellt. Ist dies der Fall, dann können Sie sich recht glücklich schätzen, denn Ihr Computer funktioniert.

Saft für unseren Mops

Für den Betrieb dieser Mikrocomputer-Minimalkonfiguration wird ein passendes Miniatur-Netzteil angeboten, das außer den benötigten 5 V/ 350 mA auch noch ±12 V/ 50 mA liefert, um bei Bedarf die serielle V24-Schnittstelle der CPU betreiben zu können. Die Schaltung sehen Sie in **Bild 7**, und **Bild 8** zeigt den zugehörigen Bestückungsplan.
Das Netzkabel wird von einer Zugentlastung gehalten, die nach unten zum Berührungsschutz mit einer Pertinax-Scheibe abgedeckt ist (**Bild 9 + 10**). Wenn Sie (nach dem

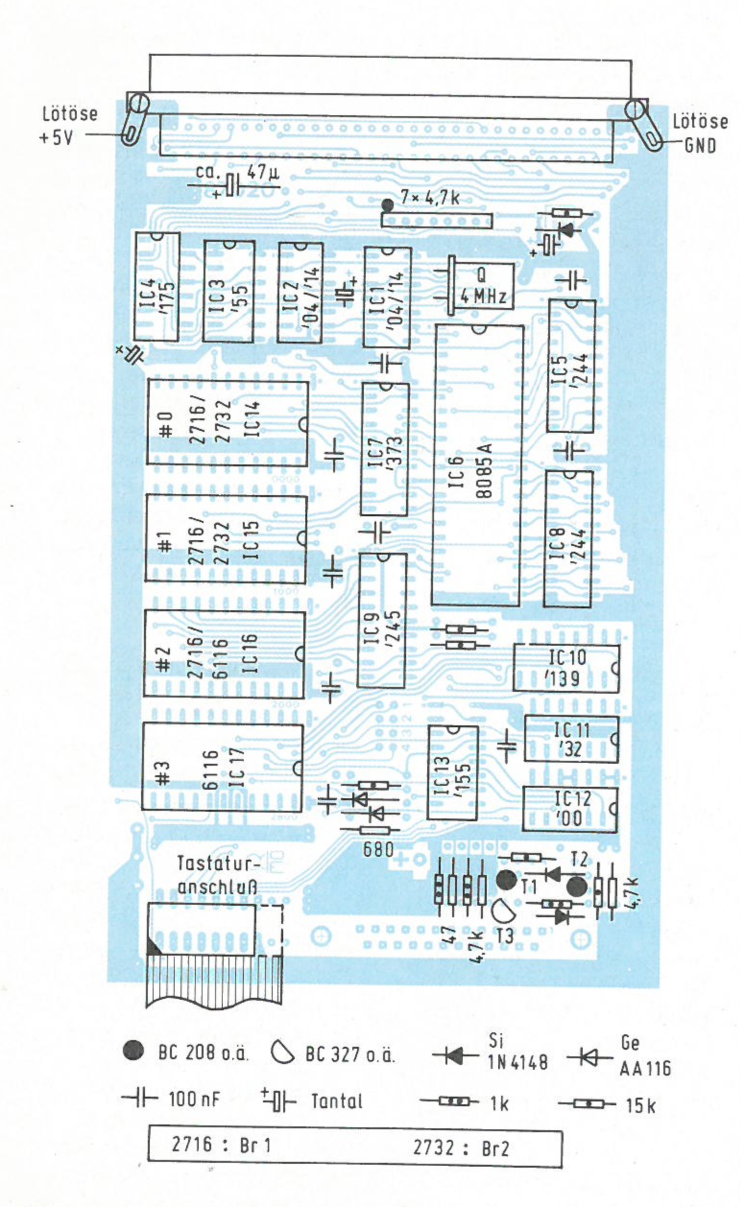


Bild 4: Bestückungsplan der CPU.

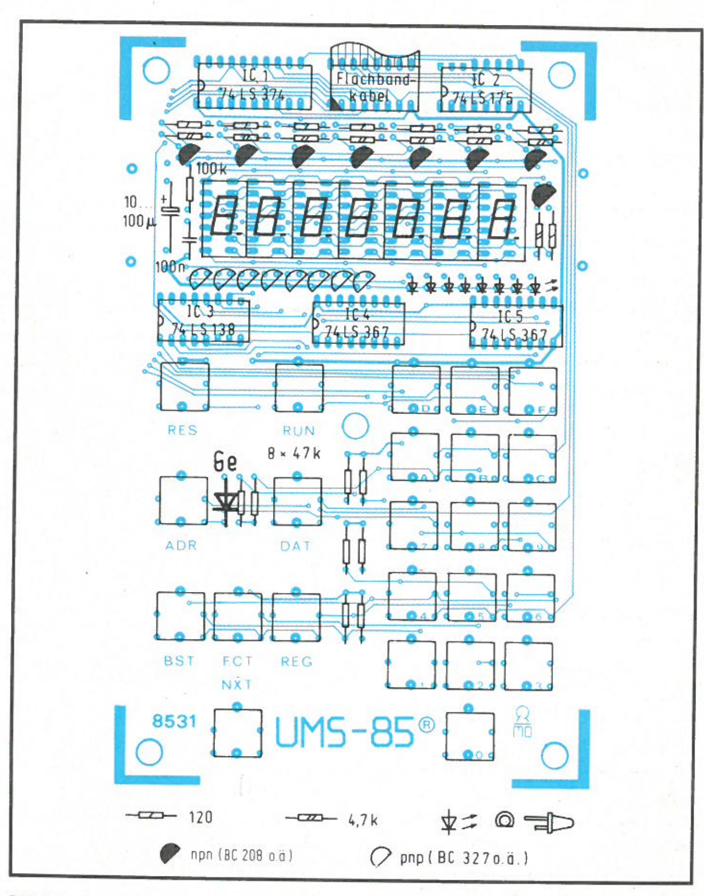


Bild 5: Schaltbild der Anzeige- und Tastenmatrix.

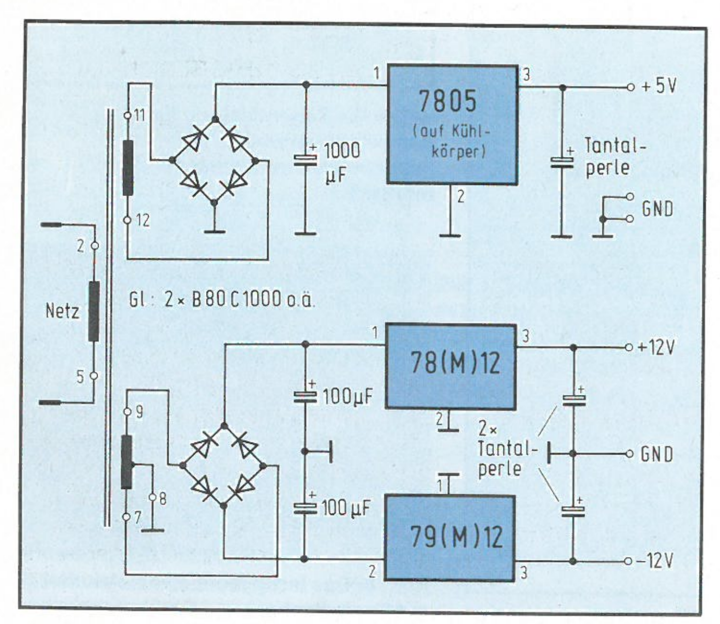


Bild 7: Ein passendes Miniatur-Netzteil zur Stromversorgung.

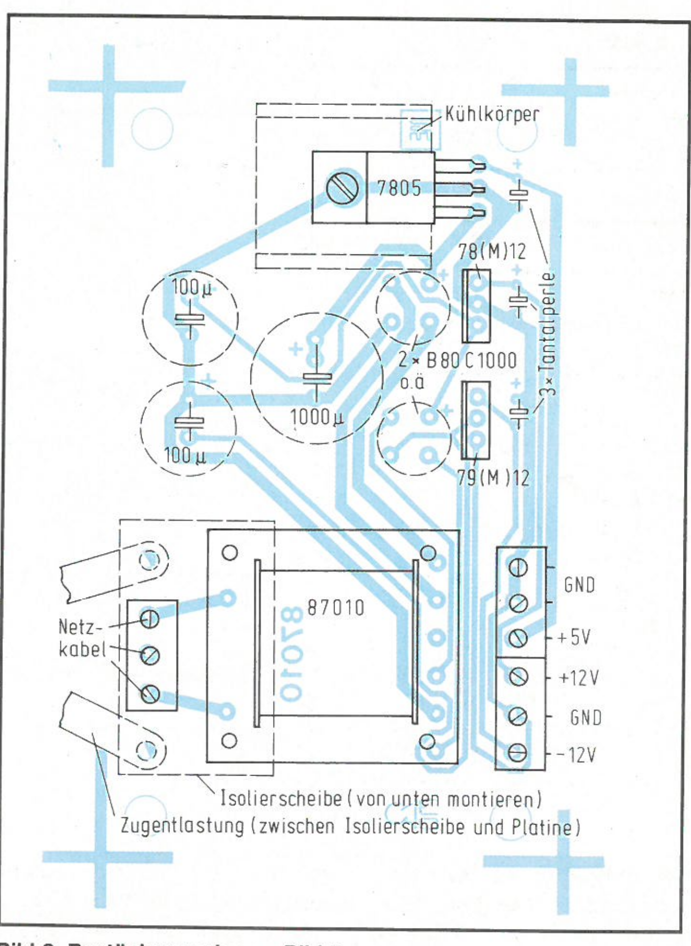


Bild 8: Bestückungsplan zu Bild 7.

Zusammenlöten) das Netzteil anschließen (bitte die Polung beachten!), dann meldet sich das fest im EPROM gespeicherte Monitor-Programm in der Anzeige, indem es linksbündig die Adresse 2800 und rechtsbündig den Inhalt dieser Speicherstelle darstellt. Die gezeigte Adresse ist die erste im RAM auf Platz 3; ab hier laden und starten Sie (in der Regel) Ihre Programme, und mittels RESET kommen Sie immer wieder an diesen Anfangspunkt zurück. Das Monitor-Programm für die Programmierung in Maschinensprache steht in drei verschiedenen Versionen zur Verfügung.

Bild 6: Bestückungsplan der Tastatur- und Anzeige-Platine.

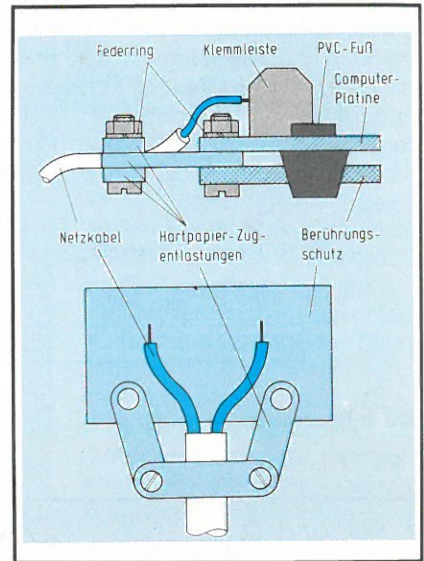


Bild 9: Die Zugentlastung für das Netzkabel wird zusammen mit dem Berührungsschutz montiert.

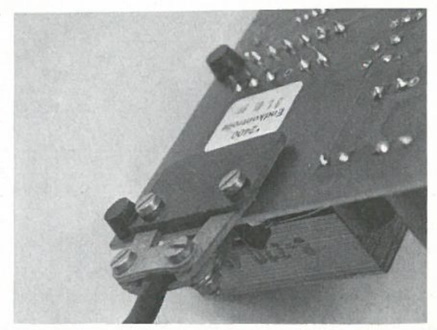


Bild 10: Das fertig montierte Netzkabel mit Zugentlastung und Berührungsschutz.

Der Monitor in drei Versionen

Nach dem Zusammenbau der Zentraleinheit und Tastatur/ Anzeige-Platine wollen Sie Ihren Computer nun natürlich auch einsetzen, und dazu müssen Sie ihm mitteilen, was er im einzelnen tun soll. Der Mikroprozessor selbst ist nicht in der Lage, irgendwie geartete Anweisungen von außen entgegenzunehmen; dazu braucht er ein eigenes Programm, das nichts weiter zu tun hat, als den Dialog zwischen Ihnen, dem Benutzer, und ihm, dem Mikrocomputer, abzuwickeln. Derartige Dienstprogramme, die im eigentlichen Sinne nichts „Richtiges" tun (wie z. B. eine Modelleisenbahn steuern), bilden die minimale Intelligenz eines jeden Systems, gleichermaßen dessen Kleinhirn (was nichts über die Leistungsfähigkeit aussagt!); man bezeichnet ein solches Grundprogramm (das auch eine Sammlung von einzelnen Programmen darstellen kann) üblicherweise als „Monitor" (Programm). Je nach Aufwand, den man da hineinsteckt, kann so ein Monitor-Programm natürlich unterschiedlich leistungsfähig sein, was sich notgedrungen auch auf den Preis bei der Entwicklung auswirkt.

Wir haben dieses System deshalb so konzipiert, daß die meisten Baugruppen (und das gilt auch für verschiedene Programme) in einer einfachen (der Basis-)Version sowie in einer luxuriösen (der Profi-) Version verfügbar sind. Damit kann auch der Hobbyist mit schmalem Geldbeutel preiswert einsteigen und seine Elektronik später nach und nach aufrüsten. Das Monitor-Programm für den MOPPEL gibt es ausnahmsweise auch noch in einer „abgemagerten" Nullversion, die all denjenigen entgegenkommt, deren Hobby-Etat gerade erschöpft ist. **Tabelle 3** gibt eine Übersicht über die drei verfügbaren Monitor-Versionen und deren Leistungsmerkmale, wozu Sie nähere Erläuterungen weiter unten finden.

Nicht selten wandelt sich ein Programm im Laufe der Zeit, d. h. es können hin und wieder Verbesserungen einfließen, so daß sich, wenn auch nur geringfügige, Unterscheidungen verschieden alter Programme ergeben können. Es ist daher außer dem Programm-Namen stets erforderlich, auch noch die jeweilige Version anzugeben. Beim MOPPEL-Monitor finden Sie beides in der Speicherzelle 003F im EPROM: Die obere Hälfte des dort angesiedelten Bytes nennt den Monitor-Typ (0: Null-, 1: Basis- und 5: Profi-Version), und in der unteren Hälfte steht die Version. Eine „51" in der Speicherstelle 003F bedeutet demnach: Profi-Monitor, Version 1.

Der Speicher: Aufteilung nach Plan

Von den 64 KBytes Adreßraum, den der 8085 bietet, können maximal 12 K auf der CPU-Platine untergebracht werden (0000...2FFF; **Bild 11**). Die oberen 2 K (2800...2FFF) sind immer mit RAM bestückt; einige dieser RAM-Plätze benötigt der Monitor für seine Aufgaben, während andere für den Anwender reserviert sind. Da sich die absoluten Adressen dieser Bereiche in den einzelnen Monitor-Ausführungen unterscheiden, geben **Tabelle 4** und die Übersicht in Bild 11 keine absoluten, sondern nur symbolische Adressen (sog. *Label*) an. Jedem Monitor mitgeliefert wird eine Symboltabelle (**Bild 12**), in der diese symbolischen Namen den jeweiligen Absolutadressen zugeordnet sind. Auf diese Weise können Sie jederzeit den Bezug zwischen einem Programm- bzw. Sprungziel-Namen und der in Ihrem

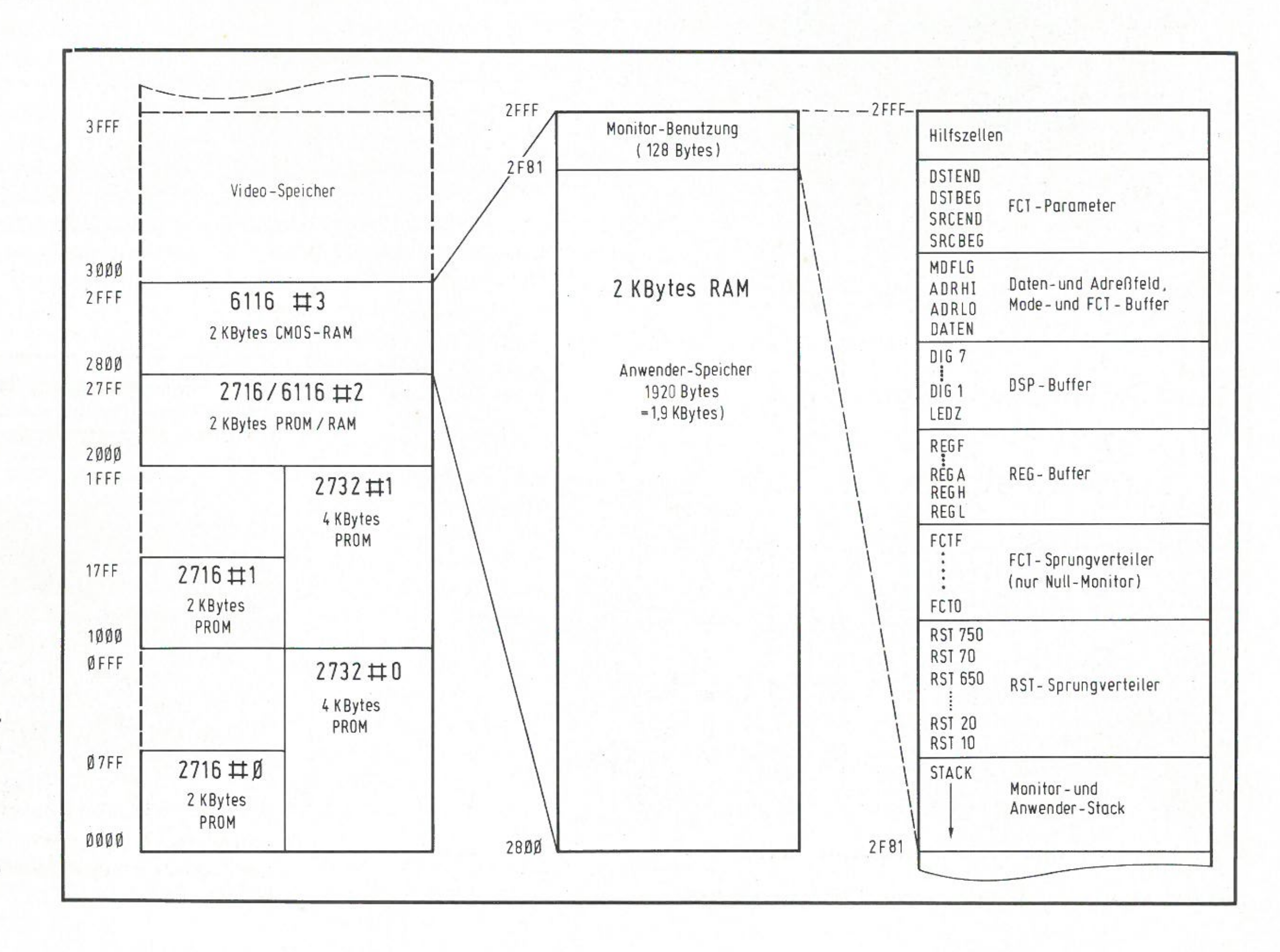


Bild 11: Der Monitor benötigt für seine Aufgaben einige RAM-Speicherstellen, deren Verteilung der Speicherplan (Memory Map) angibt.

```
MOPPEL-Null-Monitor V0.1

SYMBOL TABLE:

ADD     00FD  ADR     0277  ADR2    027D  AD
ADRLO   2FF1  ADRS    017A  BRNCH   0214  BS
CCKEY   01E9  CHECK   0040  CKEY    0201  CL
CMDKY   01DC  CNT     020A  CNVAD   011F  CC
CONVT   0116  COUNT   00CA  DABYT   014E  DA
DATN    0177  DECPT   016C  DECR    026C  DE
DELYB   02D3  DELYC   02C8  DIG1    2FE9  DI
DIG4    2FEC  DIG5    2FED  DIG6    2FEE  DI
DIGIT2  01C3  DIGIT3  01BE  DIGIT4  01B9  DI
DIGIT7  01AA  DISPL   0199  DSPFCT  032A  EN
```

Bild 12: Eine Symboltabelle stellt den Bezug zwischen symbolischen Namen (Labels) und Absolutadressen her.

EPROM gültigen Adresse herstellen.
Der RST-(RESTART)Sprungverteiler im RAM kann vom Anwender mit Sprungbefehlen geladen werden, zu denen bei Interrupts verzweigt werden soll.
Der 4-K-Bereich von 3000...3FFF wird z. Zt. noch nicht genutzt; allerdings erzeugt die CPU bereits das zugehörige CS-Signal (CS3000, Busleitung 25a). Von 4000...7FFF ist Platz für 16 K EPROM vorgesehen, wovon die unteren 8 K später mit den BASIC-PROMs bestückt werden. Ab 8000 bietet die große Speicherkarte (**Bild 13**) dann insgesamt 32 KBytes an RAM, in denen Sie sich bezüglich

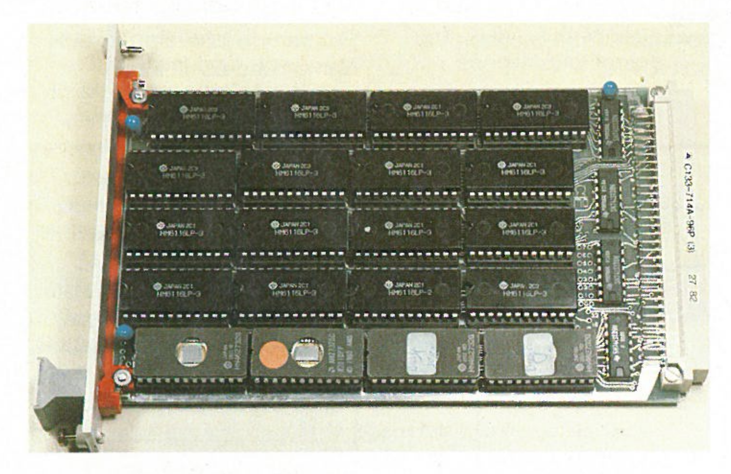


Bild 13: Dicht gedrängt sind die ICs auf der großen Speicherkarte angeordnet.

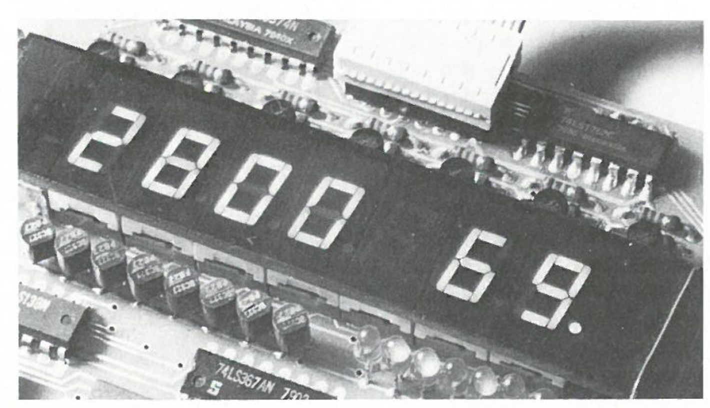


Bild 14: Der Dezimalpunkt rechts unten im Adreß- oder Datenfeld gibt an, wohin die Eingaben von der HEX-Tastatur gelangen.

Programmerstellung, Adressenverwaltung oder Heizungsoptimierung austoben können (Vorstellung später). Diese Speicherkarte läßt sich im „Banking-Verfahren" selektiv ansprechen, so daß weitere 15 solcher Karten gemeinsam an einem Bus betrieben werden können, die zusammen einen (Halbleiter-)Speicher-Bereich von insgesamt 512 KBytes (!) bieten; auch dazu folgt später mehr.

Mit Drehscheibe: Der Nullmonitor

Die Eingabe von Daten ist immer dann möglich, wenn der Dezimalpunkt rechts unten im Datenfeld der Anzeige aufleuchtet (**Bild 14**); das ist nach Betätigen der **DAT**-Taste (und nach jedem Rücksetzen über **RES**) der Fall. Leuchtet (nach Betätigen der **ADR**-Taste) der Dezimalpunkt unten rechts im Adreßfeld der Anzeige auf, gelangen die Eingaben eben dorthin. Eingegebene und im Datenfeld angezeigte Daten werden unter der links erscheinenden Adresse abgelegt, *nachdem* **NXT** gedrückt und losgelassen worden ist; gleichzeitig wird die Adresse um Eins erhöht. Betätigt man **BST** (*Back-Step* = Schritt zurück), wird die Adresse um Eins erniedrigt, und der Inhalt der nun adressierten Zelle erscheint rechts im Datenfeld. Mittels **RUN** startet man seine eigenen, zuvor im RAM abgelegten Programme, und zwar geschieht dies immer von derjenigen Adresse aus, die zum Zeitpunkt der RUN-Auslösung im Adreßfeld steht.
Mit der Taste **REG** lassen sich die internen CPU-Register laden und inspizieren, indem man außer REG noch den gewünschten Register-Namen eingibt (z. B. „A" oder „E"). Die anschließende Dateneingabe, abgeschlossen durch NXT, lädt das zugehörige Register, und mit diesem vom Anwender eingegebenen Wert beginnt später die Programmausführung des Benutzerprogramms.
Achtung! Nach dem Laden eines Registers darf bis zum

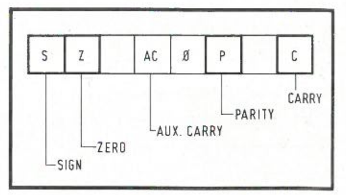


Bild 15: Vier Zustandssignale (Flags) im Register F können über die bedingten Sprungbefehle „abgefragt" werden.

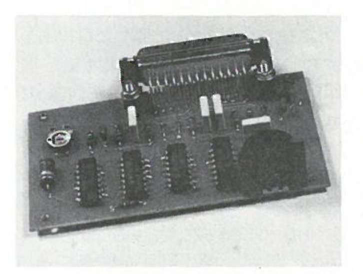


Bild 16: Die CPU-Karte steuert über den seriellen Ein-/Ausgang das Cassetten-Interface an.

Starten des Programms nicht RES betätigt werden! Zur Startadresse des Anwenderprogramms gelangt man in diesem Fall über ADR, gefolgt von der Adreßeingabe selbst. Beendet man ein laufendes Anwenderprogramm per RES, kann man sich anschließend den Inhalt aller Register ansehen, den sie zum Zeitpunkt der RES-Betätigung gerade hatten. Auf diese Weise kann man sich auch die Zustandssignale (*Flags*) der CPU ansehen, die nach REGF gerettet werden (**Bild 15**); laden läßt sich dieses Register vor dem Start natürlich nicht, weil nur die CPU die FLAG-Bits beeinflussen kann, und das geschieht ausschließlich *nach* einer arithmetisch-logischen Operation.
Die bis hierhin genannten Eigenschaften besitzen alle Monitor-Ausführungen gleichermaßen; nur die besondere Funktion der **FCT**-Taste (Funktionserweiterung) unterscheidet sich beim Nullmonitor von der der größeren Brüder: Wenn man FCT drückt, gefolgt von einer der 16 Tasten 0...F, springt der Monitor an diejenige Stelle ins RAM, die zu der betreffenden Eingabe gehört (also „FCT-0"...„FCT-F"). Das sind 16 × 3 Bytes, die Sie selbst mit 16 Sprungbefehlen (Dreiwort-Be-

fehl JMP XXYY) belegen können, um per Tastendruck zu einem bestimmten Ziel Ihrer Wahl verzweigen zu können (Sprungverteiler). Auf diese Weise haben Sie (in der Monitor-Nullversion) 16 Tasten, die nach landläufigem Sprachgebrauch „frei programmierbar" sind.

Funktionen mit Komfort

Wie aus Tabelle 2 ersichtlich ist, bietet der **Basis-Monitor** eine Reihe weiterer Funktionen, die über die Funktionstaste FCT aufgerufen werden (die freie Belegbarkeit entfällt hier). Beim Druck auf FCT werden das Adreßfeld auf „0000" und das Datenfeld auf „FF." gesetzt. Der Monitor erwartet nun die Spezifikation der gewünschten Funktion, was durch Eingabe von 0, 1, 4, A, C oder D geschieht; danach steht im Adreßfeld „F0...FD", je nachdem, welche Funktionen Sie gewählt haben (die übrigen Funktionen 2, 3, 5...9 sowie B und E sind beim Basis-Monitor nicht belegt). Anschließend erfolgt die Eingabe von Adressen, die automatisch (also ohne den Druck auf ADR) ins Adreßfeld gelangen. Sind mehrere Adreßangaben erforderlich (s. u.), werden sie durch NXT getrennt. Die Ausführung der gewählten Funktion beginnt mit dem Loslassen der aktivierten RUN-Taste.

Das Programm **DIGUR** (FCT-4) stellt ein einfaches Demonstrationsbeispiel für eine Digitaluhr dar, die in der Anzeige im Sekundentakt abläuft. Das Starten erfolgt mittels FCT-4, gefolgt von der Eingabe der Stunden und Minuten, die mit RUN abgeschlossen wird (Sekunden werden intern auf Null gesetzt); danach „tickt" der Mops treu vor sich hin, was, wie gesagt, nur anschaulichen Charakter besitzt und nicht etwa eine genaugehende Quarzuhr ersetzen soll.

Mit **SEROT** (FCT-0) und **SERIN** (FCT-1) wird die Daten-Aus- bzw. -Eingabe mittels Cassetten-Interface (separate Baugruppe; **Bild 16**) verwaltet. Dazu muß nach Eingabe von FCT-0 bzw. FCT-1 erst die Startadresse, dann NXT und danach die *Länge* des zu übertragenden Speicherbereichs eingegeben werden; die Übertragung selbst wird wiederum mit RUN ausgelöst.

Als Magnetbandgerät eignen sich preiswerte, handelsübliche Recorder, nur bei der Wahl der Bandsorte sollte man spendabler sein: Bitte verwenden Sie nicht zu dünnes Band (C60) und nehmen Sie eine gute (Hi-Fi-)Qualität.

Mit FCT-C (= **COPY**) können Sie Speicherbereiche verschieben, und zwar auch um nur *eine* Adresse vor- bzw. rückwärts (viele Monitor-Programme können diesen überlappenden Kopierbetrieb nicht!). Dazu geben Sie bei der Adreßspezifikation zunächst die Startadresse (plus NXT) ein, dann die Endadresse (wiederum mit NXT) und schließlich die Zieladresse, von der an der Speicherblock abgelegt werden soll. Mit RUN beginnt der Transfer, und anschließend zeigt der Monitor sofort die Zieladresse, damit Sie sich davon überzeugen können, daß die Übertragung einwandfrei geklappt hat.

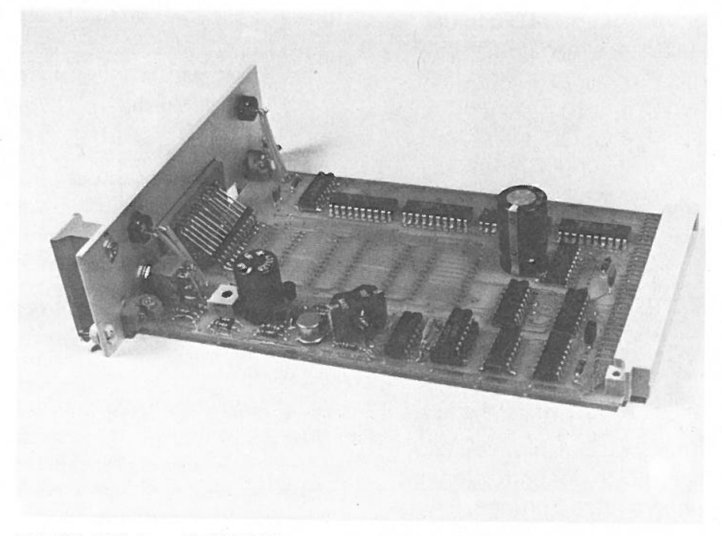


Bild 18: Mit dem EPROM-Programmierzusatz kann der Anwender die gängigen EPROM-Typen selbst programmieren.

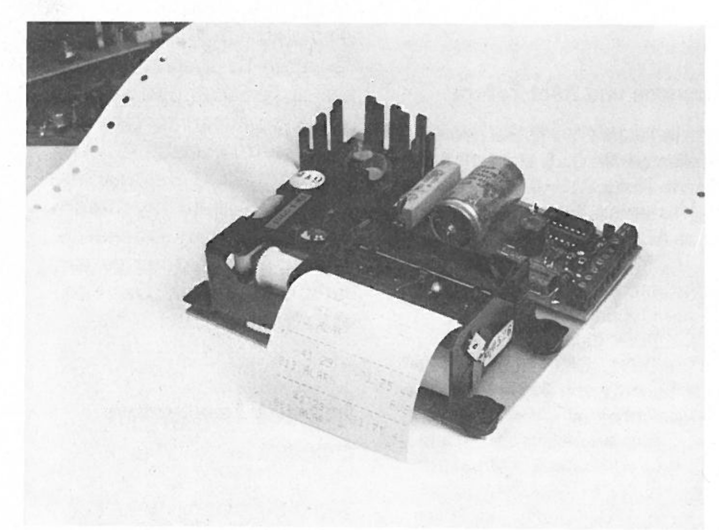


Bild 17: Für Speicherabzüge, Protokolldrucke und Klartextbelege läßt sich der Thermodrucker einsetzen.

Ähnlich läuft es mit FCT-A (= Auffüllen; **FILL**) ab; hiermit schreiben Sie einen bestimmten Speicherbereich mit einer Konstanten (z. B. „00" beim Löschen) voll. Es werden wiederum Start- (mit NXT) und Endadresse (auch gefolgt von NXT) eingegeben, und als dritte Eingabe müssen Sie noch (zweistellig) „sagen", was Ihr Mops in den genannten Bereich einschreiben soll. Erst nach dieser dritten Eingabe können Sie die Auffüll-Prozedur über RUN starten, und in Windeseile steht in dem von Ihnen genannten Bereich eine Latte der zuletzt angegebenen Konstanten.

Wenn Sie FCT-D (Drucken; **DUMP**) anwählen, können Sie sich mit dem Thermodrucker (separate Baugruppe; **Bild 17**) einen Speicherbereich ausdrucken lassen; dazu geben Sie, wie eben beschrieben, wiederum Start- und Endadresse ein (jeweils gefolgt von NXT) und sagen dem Computer dann, ob der Ausdruck hexadezimal (mit vorangestellter Adresse) oder im Klartext (in Form von ASCII-Zeichen) erfolgen soll: Bei hexadezimalem Druck geben Sie als dritten Parameter „EE" ein (zweistellig genügt), während der ASCII-Ausdruck als dritte Eingabe ein „AA" erfordert. Über den letztgenannten Weg ist beispielsweise der Ausdruck von Texten möglich, angefangen bei Visitenkarten bis hin zu Schulaufsätzen (Schreibbreite: 20 Stellen). Im HEX-Mode werden immer Adresse und mindestens acht Bytes ausgegeben, während im ASCII-Mode mindestens eine ganze Zeile mit 20 Zeichen gedruckt wird.

Schritt für Schritt: Der Single-Step

Im Profi-Monitor dient die Funktion FCT-B (Breakpoint; **BREAK**) dazu, im Programm bestimmte Haltepunkte zu setzen; wenn man beispielsweise an einer bestimmten Stelle im Programm nachsehen möchte (etwa in den Registern), ob alles nach Plan verläuft, läßt man die Programmausführung dort stoppen und schaut nach. Auf diese Weise lassen sich eigene Programme besonders effektiv testen und Fehler aufspüren. Die Eingabesequenz hierfür lautet FCT-B, gefolgt von der Start-

adresse (plus NXT) und der Halteadresse; mit RUN läuft das Programm nun bis zum angegebenen Programmpunkt, und dort können Sie mit Hilfe der übrigen Monitor-Funktionen weitermachen.
In dieselbe Richtung, nur noch etwas verfeinert, zielt der Einzelschritt-Betrieb (= FCT-E; **SSTEP**); hier brauchen Sie nur die Startadresse einzugeben, von der aus Sie „lossteppen" wollen, und mit jedem RUN geht das Programm dann genau einen Befehl weiter (einschließlich der bedingten und unbedingten Sprünge, CALLs und RETURNs). Sie haben damit die Möglichkeit, Ihre Programme bei der Ausführung Schritt für Schritt zu verfolgen und dabei zu sehen, wo sie „aushaken".
Achtung! Der Einzelschritt-Betrieb läuft nur mit Programmen im RAM ab; Unterprogramme werden in beliebigen Ebenen verarbeitet (auch nested Subroutines).
Für Profis unerläßlich ist die Möglichkeit, selbst EPROMs zu programmieren. Dazu ist die entsprechende Baugruppe erforderlich (**Bild 18**), die folgendermaßen angesteuert wird: FCT-6 (**WRPROM**), Start- und Endadresse des Quellbuffers im RAM, dann Startadresse im EPROM (jeweils durch NXT getrennt), und zum Schluß geben Sie als vierten Parameter ein, ob es sich um ein 2758 bzw. 2716 („16") oder ein 2732 („32") handelt. Mit RUN beginnt der Spannungswandler auf der Programmierkarte seine Arbeit (er macht aus den 5 V der Systemversorgung die erforderlichen 26 V für die Programmierspannung), und eine Sekunde später beginnt die eigentliche Programmierung (pro Byte dauert sie 50 ms), die für ein „ganzes" 2716 rund 1,5 Minuten und für die 4096 Bytes eines 2732 etwa drei Minuten in Anspruch nimmt. Wer über diese „endlos" langen Zeiten stöhnt, der sollte selbst einmal versuchen, in dieser Zeit über 32 000 Bits dauerhaft irgendwo einzuhämmern! Das Lesen eines in die Programmierkarte eingesetzten EPROMs geschieht ganz analog per Aufruf FCT-7 (**RDPROM**), gefolgt von den drei Adreßangaben und dem PROM-Typ „16" bzw. „32".

Tabelle 3. Eigenschaften der drei Monitor-Versionen

	PROMO Profi-Monitor	MOBAS Basis-Monitor	MONUL Null-Monitor
RES	Hardware-RESET		
ADR	Adreß-Mode wählen		
DAT	Daten-Mode wählen		
REG	Register-Mode wählen		
NXT	Adresse/Register erhöhen und Daten einschreiben		
BST	Adresse/Register erniedrigen		
RUN	Programm starten		
FCT-0	SEROT		frei belegbar (Sprungziele FCT-0... FCT-F im RAM)
FCT-1	SERIN		
FCT-2	V24 OUT	–	
FCT-3	V24 IN	–	
FCT-4	DIGUR		
FCT-5	SSTEP	–	
FCT-6	WRPROM	–	
FCT-7	RDPROM	–	
FCT-8	WRITUR	–	
FCT-9	READUR	–	
FCT-A	FILL		
FCT-B	BREAK	–	
FCT-C	COPY		
FCT-D	DUMP		
FCT-E	SSTEP	–	
FCT-F	nicht belegt		

```
Fr,06.08.82;16:23:23h
```

Bild 19: Die von der Echtzeit-Uhr gelieferten Daten können für die Zeiterfassung oder für Protokolldrucke verwendet werden.

Tabelle 4. Einige Monitor-Unterprogramme und RAM-Zellen.

HHKEY	Abfrage der HEX-Tastatur; keine Taste: CY = 0. Bei gedrückter Taste ist CY = 1 und die Tasten-Nr. 0...F steht im ACC
CCKEY	Abfrage der Befehlstasten; keine Taste: CY = 0. Bei gedrückter Taste ist CY = 1 und die Tasten-Nr. 0...6 steht im ACC
BYTE	Siebensegmentumsetzung des ACC; in D, E muß Zieladresse für SSG-Code stehen
SSGCV	Siebensegmentumsetzung des unteren ACC-Nibbles; in D, E muß Zieladresse für SSG-Code stehen
SSGTB	Siebensegmenttabelle; Balkenmuster für 0...F
DISPL	Überschreibt den Display-Buffer DIG1...DIG7 + LEDZ in die Anzeige plus LED-Zeile; feste Laufzeit von 8 ms
COMPAR	Vergleicht H, L mit D, E; bei Gleichheit ist CY = 0
ONSEC	feste Laufzeit von 1 s
DELAY	feste Laufzeit von 100 ms
DELYB	feste Laufzeit von (B) * 1 ms
DATEN	Datenfeld
ADRLO ADRHI	Adreßfeld (untere und obere Hälfte)

Stets parat: Uhrzeit und Datum

Sie können sich drehen und wenden, wie Sie wollen: Ein Computer ohne eingebaute Uhr ist höchstens ein halber Computer. Nicht nur zum Ausdruck von Datum und Uhrzeit auf Briefen oder bei der Zeiterfassung (**Bild 19**), sondern etwa auch zur rechtzeitigen Mahnung nahender Geburts- und Gedenktage eignet sich ein Mops bestens, sofern er „weiß", was die Stunde geschlagen hat. Und genau hierfür vorgesehen ist die Baugruppe der Echtzeit-Uhr (**Bild 20**), die auch bei abgeschalteter Systemversorgung monatelang vor sich hin tickt, wenn sie mit einem kleinen Akku gepuffert wird.
Mit FCT-9 (= **READUR**) lassen sich Uhrzeit und Datum von dieser Baugruppe auslesen und von der zusätzlich angegebenen Adresse an ins RAM überschreiben; bei angeschlossener HEX-Tastatur erscheint die Uhrzeit außerdem noch in der Anzeige (**Bild 21**; diese Uhr „geht", im Gegensatz zu der oben beschriebenen Demo-Uhr, ganz genau). Über die Funktion FCT-8 (**WRITUR**) läßt sich die Uhr stellen; dazu geben Sie (wie sonst die Adressen) zunächst Stunden und Minuten ein (z. B. 1745 für 17.45 Uhr), dann den Wochen- und Kalendertag (für Montag 01 usw.; z. B. 0408 für Do, den 8.) und schließlich folgen Monat und Jahr (z. B. 0882 für August '82). Mit RUN wird diese Uhrzeit in das interne Uhren-IC überschrieben, wobei die Sekunden wiederum automatisch auf Null gesetzt werden; anschließend tickt der CMOS-Baustein (fast) bis zum jüngsten Tag, sofern er immer mit mindestens 2 V und ein paar Mikroampere versorgt wird (vgl. „Das interessante IC in ELO 8/82).

Immer langsam voran

Wenn Sie hier mit geballten Monitor-Funktionen förmlich überschüttet worden sind,

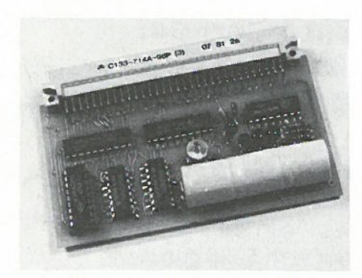


Bild 20: Bei der Echtzeit-Uhr sorgt ein kleiner Akku dafür, daß das Uhren-IC auch bei abgeschalteter Systemversorgung weiterläuft.

dann soll das nicht heißen, daß Sie all das nun etwa auswendig hersagen sollen. Erstens muß man die Eingabesequenzen einmal selbst praktisch nachvollzogen haben, um anschaulich zu sehen, „wie der Hase läuft". Und zweitens gehen wir auf die hier vorgestellten Baugruppen und Programmteile noch ausführlich ein. Bei späteren Unklarheiten finden Sie hier aber alles komprimiert zusammengefaßt, was Ihnen lästiges Suchen ersparen soll. Und zum Schluß dieser Vorstellung noch eins: Die Erläuterungen der Funktionen 2,3 und 5 sind hier nicht etwa vergessen worden; sie sind für späteren Gebrauch reserviert.

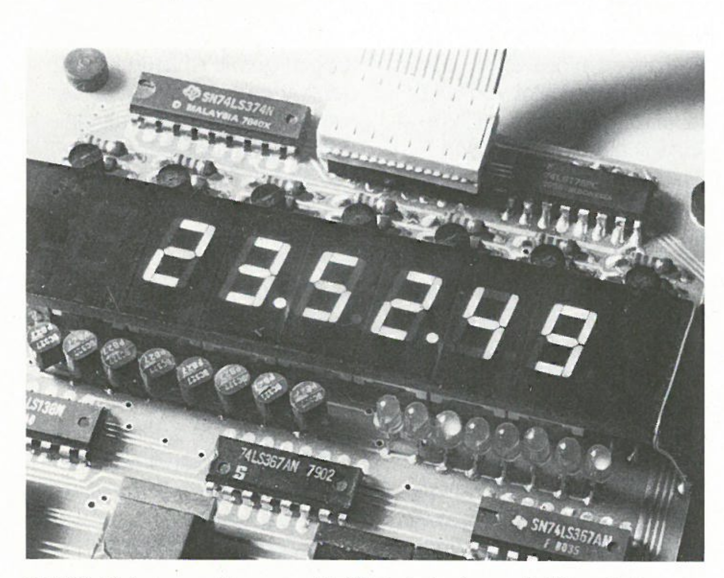


Bild 21: Bei angeschlossener HEX-Tastatur kann die Uhrzeit der Echtzeit-Uhr in die Anzeige überschrieben werden.

Eine Behausung für MOPPEL

Der Bus verbindet sie alle

Die verschiedenen Erweiterungskarten des Systems finden Platz in einem 483-mm-(bzw. 19-Zoll-)Einschubrahmen, der als Rückwandverdrahtung die Busplatine mit den einzelnen Steckplätzen besitzt (**Bild 22**). Der Platz ganz links ist für das große Netzteil reserviert, und rechts außen sind zwei Leisten mit je 24 Stiften für den Anschluß von Peripheriegeräten vorgesehen. Alle anderen neun Steckplätze sind gleichberechtigt und demzufolge parallel an den Bus angeschlossen (vgl. Tabelle 2). Das Layout der Platine finden Sie auf dieser Seite (**Bild 23**), und bei der Bestückung sollten Sie mit wachen Augen zu Werke gehen, damit bei dieser scheinbar problemlosen Arbeit nichts schiefgeht; also bitte erst lesen, dann löten! Die 64poligen VG-Buchsenleisten sind so einzulöten, daß die Aussparungen an der oberen und unteren Schmalseite alle in Richtung der 24poligen Stiftleisten zeigen. Die Feinsicherung kommt auf die Bauteilseite (**Bild 24**), während die drei Buchsen für Drucker-, TV-Monitor- und Bandgerät-Anschluß auf der Lötseite zu montieren (und zu verlöten) sind (**Bild 25**). Die Massefahne der Cinch-Buchse wird mit einem durchgesteckten Drahtende mit der Platinen-Masse verbunden, während vom Innenleiter dieser Buchse eine Drahtverbindung zum Punkt 1 herzustellen ist (**Bild 26**). Die Lötfahnen der Tonbandbuchse sollten Sie auf der Bestükkungsseite unbedingt umbiegen, weil die beim Herausziehen auftretenden Kräfte sonst die Leiterbahnen ablösen; der zusätzlich eingelötete Draht (von Stift 2 zur Massefahne) erhöht die mechanische Festigkeit noch weiter. Die drei übrigen Buchsen sind für besondere Einsatzfälle vorgesehen; sie werden normalerweise nicht bestückt. Bitte achten Sie beim Löten peinlichst darauf, daß keine unbeabsichtigten Lötbrücken zum Nachbar-Pin entstehen. Nach dem Einlöten einer Buchsenleiste sollten Sie zum schnellen Testen Ihre MOPPEL-CPU aufstekken und sehen, ob noch alles funktioniert. Ist das nämlich nicht der Fall, haben Sie einen Lötklecks fabriziert, den Sie aber relativ schnell finden, weil er nur an der zuletzt eingelöteten Leiste passiert sein kann. Wer unbedingt alle Plätze auf einmal bestücken muß, hat bei einer späteren Fehlersuche wesentlich mehr Freude.

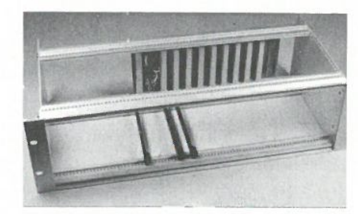


Bild 22: Zur Verbindung der Platinen untereinander dient die als Rückwandverdrahtung eingesetzte Bus-Platine.

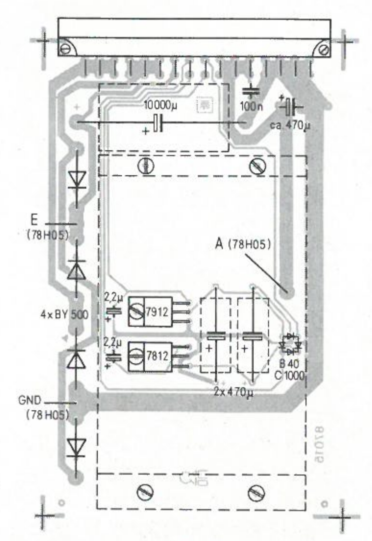


Bild 23: Die große Netzteilkarte liefert außer +5 V noch +/−12 V zur Versorgung der V24-Schnittstelle.

Ein Kartenhaus, das nicht einstürzt

Der Zusammenbau der Einschub-Mechanik sollte auch für einen Elektriker kein unüberwindbares Hindernis sein. Am besten beginnen Sie mit der oberen und unteren Frontleiste, in die zunächst die beiden Gewindeschienen eingeschoben werden; zusammen mit einem Griff-Seitenteil werden sie mit einem Seitenteil verschraubt (**Bild 27**). Die beiden hinteren Querstreben schrauben Sie in dem Abstand an, der Ihnen von den Rastnasen der Leiterplatten-Führungsschienen vorgegeben wird. Mit der linken Seitenwand verfahren Sie analog, und die sogenannten Z-Schienen zur Befestigung der VG-Leisten schrauben Sie von hinten an (es sind bewußt keine Querstreben mit integrierter Z-Schiene verwendet worden). Die Busplatine betten Sie nun so in den Rahmen ein, daß das Befestigungsloch der Netzteil-Leiste 142,5 mm von der linken Seitenwand entfernt liegt; zum Anflanschen an die Z-Schienen muß nicht jede VG-Leiste angeschraubt werden, jede zweite oder dritte

genügt vollkommen. Des MOPPELs Behausung wird komplettiert, indem Sie die Plastik-Führungsschienen einklinken, und zwar mit der dikken Seite wieder in Richtung der 24poligen Stiftleisten (**Bild 28**). Das Gefühl, das einem beim ersten Einschieben einer Europakarte beschleicht, wird Sie die Brust bis zum Bersten schwellen lassen: MOPPEL wird zum richtigen Computer, den Sie mit einigem Geschick sogar selbst zusammenbauen können!

Jetzt fließt der Saft in Strömen

Aus der Schaltung des Netzteils (**Bild 29**) ersehen Sie, daß es sich hierbei um längsgeregelte Stabilisierungen handelt. Der 5-V-Festspannungsregler nimmt die Oberspannung vom 10 000-µF-Lade-Elko ab; zur Abfuhr der entstehenden Verlustwärme thront er auf einem überdimensionalen Kühlkörper (**Bild 30**), bei dem das Aufschrauben derart erfolgen muß, daß die Anschlußstifte nicht mit dem darunter vorgesehenen Elko in Konflikt kommen; drei dicke Drähte verbinden den Regler mit der Platine (**Bild 31**). Unter dem Kühlkörper ist Platz für zwei weitere Regler vorgesehen, die +/−12 V bereitstellen. Für den reinen MOPPEL-Betrieb sind sie nicht erforderlich, wohl aber zur Versorgung der V24-Schnittstelle (z. B. für den Anschluß eines großen Druckers). Den Trafo können Sie im Prinzip plazieren, wohin Sie wollen; eine recht elegante Möglichkeit stellt das rückwärtige Anschrauben an die Z-Schienen dar, das Sie auf **Bild 32** erkennen. Die 9-V-Wicklung des Trafos (Lötösen 13 und 14) verbinden Sie bitte mit den Punkten 11 und 12 der Bus-Platine (Polung beliebig, es handelt sich um Wechselspannung). An die Punkte 21 und 22 der Busplatine kommt die Spannung „0" und „23" (Trafo-Lötösen 8 und 10), und den Mittelpunkt „11,5" (Lötöse 9) verbinden Sie bitte mit Masse (vgl. auch Schaltbild). Die 18-V-Wicklung des Trafos (Lötösen 11 und 12) bildet schließlich die Oberspannung für den Thermodrucker; sie muß an die beiden untersten Lötaugen der 24poligen Stiftleisten führen („30"); das freie Auge „8" unmittelbar unter der Masseleitung verbinden Sie abschließend bitte noch mit dem Anschluß „8" am Netzteil-Steckplatz der Bus-Platine (ungeregelte 8-V-Spannung für den Drucker-Motor). Wenn Sie die Feinsicherung eingesetzt haben, können Sie die Netzteilkarte und die Zentraleinheit einstecken und den Aufbau einweihen, indem Sie den Trafo primärseitig ans Netz anschließen.

Bitte isolieren Sie die netzspannungsführenden Lötanschlüsse auf das Sorgfältigste!

Wer unbedingt will, kann zusätzlich einen Netzschalter vorsehen. Notwendig ist der nicht, denn in der Regel sind zusammen mit dem Computer weitere Geräte ein- bzw. auszuschalten (z. B. TV-Monitor und Drucker), was sich am einfachsten über eine schaltbare Verteilerdose bewerkstelligen läßt (**Bild 33**). Auf jeden Fall sollten Sie es vermeiden, einen solchen Schalter an der Frontseite anzuordnen, weil dann die 220-V-Leitungen mitten in der Elektronik umherirren und unübersehbaren Schaden anrichten können, wenn der Fall der Fälle einer mangelhaften Isolation eintritt! Es hat sich als vorteilhaft erwiesen, die Netzverbindung über eine Kaltgeräteschnur herzustellen, die bei Bedarf vom Gerät selbst getrennt werden kann.

Achten Sie peinlichst darauf, sowohl den Rahmen als auch den Trafo mit dem grün-gelben Schutzleiter zu verbinden!

Diese Hinweise sind an sich eine Selbstverständlichkeit, und Sie sollten sie nicht nur zur Kenntnis nehmen, sondern auch wirklich in die Tat umsetzen!

Der Weg zur Programmbibliothek

Um Programme sammeln und archivieren zu können, verfügt MOPPEL über zwei Magnetband-Interfaces. Eins können Sie über den 25poligen Subminiaturstecker mit der seriellen Schnittstelle der CPU verbinden (**Bild 34**), was aber auch über vier einfache Drahtbrücken erfolgen kann. Die Monitor-Funktionen FCT-0 (Cassetten-Output) bzw. FCT-1 (Cassetten-Input) steuern diese Interface-Karten an, wobei Sie nach Eingabe der Funktions-Nummer noch die Startadresse des Quell- bzw. Zielbuffers (gefolgt von NXT) sowie die Blocklänge (Byte-Anzahl) eingeben müssen.
Das Ausgeben bzw. Einlesen starten Sie dann per RUN (vgl. auch Vorstellung des Basis-Monitors).
Das Aufzeichnungsverfahren arbeitet nach dem Prinzip der Frequenzumtastung (LOW-Bit = niedrige und HIGH-Bit = hohe Frequenz) mit einer Übertragungsrate von 250 Bits pro Sekunde. Zum Abgleich muß die Detektionsschaltung mit Hilfe eines Potis so eingestellt werden, daß die vom Monoflop erzeugte Zeitspanne genau in der Mitte der Periodendauer von niedriger und hoher Aufzeichnungsfrequenz liegt. Das hört sich unheimlich kompliziert an, ist es aber in Wirklichkeit gar nicht, und mit einem winzigen Programmchen können Sie (ohne Oszilloskop) Ihren MOPPEL zu dieser Zeitmessung umfunktionieren. Die entsprechenden Anweisungen liegen jedem Bausatz bei.

Achtung! Auf der Platine des Cassetten-Interfaces werden +12 V und +5 V sowie −12 V und Masse kurzgeschlossen, um in der Minimalkonfiguration mit einfacher 5-V-Versorgung auszukommen; die beiden +/−12-V-Regler auf der Netzteilkarte dürfen beim Betrieb dieser Karte folglich nicht eingesetzt werden (oder Sie trennen die betreffenden Leiterbahnen auf der CPU-Karte auf).

Ein Eldorado: Die 48-KByte-RAM/ROM-Karte

Bereits bei der Vorstellung der drei Monitor-Versionen haben wir Ihnen angekündigt, daß der vom 8085 gebotene Adreßraum von 64 KBytes beim MOPPEL lückenlos gefüllt wird. Die ersten 16 K belegen CPU und Video-Speicher, die nächsten 16 K sind für EPROMs vorgesehen, und die gesamte obere Hälfte des Adreßraums können Sie mit RAMs füllen; die letztgenannten 48 KBytes finden komplett auf der großen Speicherkarte Platz, die damit bis zum Stehkragen gefüllt ist (**Bild 35**). Organisation und Aufbau dieser Karte sind ein Muster an Ordentlichkeit. Das beginnt bei der physikalischen Anordnung der Speicherbausteine, die in aufsteigender Reihenfolge angeordnet sind, beginnend rechts unten und sich dann bis nach links oben rankend (**Bild 36**). Jeweils neben dem Anschlußstift 1 steht auf der Platine das oberste Digit des Adreßbereichs, der in dem betreffenden Speicher untergebracht ist, also z. B. „4" für den 4000er-Adreßraum usw. Natürlich führen die acht Datenbits und die untersten 11 Adreßbits parallel an alle 20 Speicherbausteine; aus diesem Grund verzichten wir auch auf einen kompletten Detailschaltplan, der ohnehin nur 20 Kästchen mit (nahezu) derselben Beschaltung zeigen würde. Viel wesentlicher ist die Decodierlogik, die für jedes einzelne der 20 Speicher-ICs ein eigenes Chip-Select-Signal erzeugt (**Bild 37**). Die Karte kann über verschiedene Brücken sehr vielseitig an unterschiedliche Einsatzfälle angepaßt werden. Um Sie hier nicht zu verwirren, wollen wir diese Feinheiten zunächst hintenan stellen.
Mit den beiden obersten Adreßbits A14 und A15 werden über IC4/„ obere Hälfte" (d. h. der im Bild oben gezeichnete Decodierer) vier Blöcke zu je 16 K angesprochen (CS0...CSC), wovon die drei oberen Blöcke (ab 4000,

ab 8000 und ab C000) auf der Karte selbst angeordnet sind. Davon werden die ersten 16 K mit vier EPROMs vom Typ 2732 belegt; diese spricht IC4/ untere Hälfte an (CS4000, CS5000, CS6000 und CS7000). Für die nächsten 32 K sind 16 Chip-Select Signale erforderlich, weil RAMs mit einer Kapazität von 2 KBytes eingesetzt werden; je acht dieser Selektierungssignale erzeugen die Decodierer IC1 und IC2, die nach einer übergeordneten Vorauswahl durch IC4/ oben freigegeben werden.

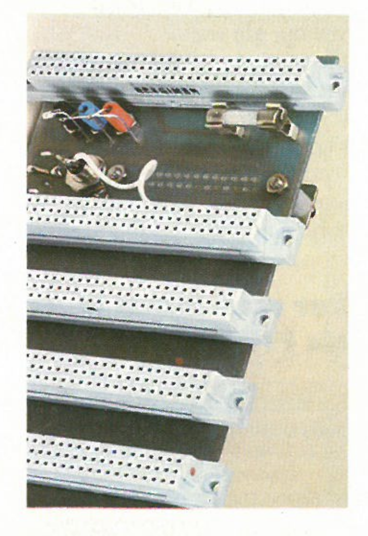


Bild 24: Die Feinsicherung für die +5-V-Leitung wird auf der Bauteilseite eingelötet. Unbedingt auf richtige Ausrichtung der 64poligen Buchsenleisten achten: Die kleinen Aussparungen oben und unten müssen nach rechts (in Richtung der 24poligen Stiftleisten) zeigen.



Bild 25: Zum Anschluß der verschiedenen Peripheriegeräte sind auf der Bus-Platine die entsprechenden Steckbuchsen vorgesehen.

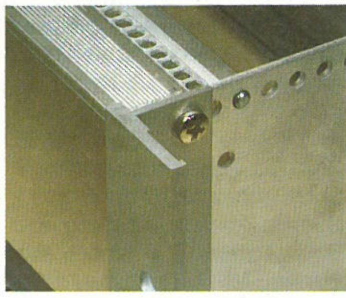


Bild 26: Zur Verbesserung der mechanischen Festigkeit sollten die Anschlußstifte der Tonbandbuchse rückwärtig umgebogen und durch eine zusätzliche Drahtverbindung gesichert werden.

Bild 27: Der Zusammenbau der Einschub-Mechanik ist dank der vorgefertigten Teile ein Kinderspiel.



Bild 28: Die Platinen werden von Plastik-Schienen geführt und gehalten.

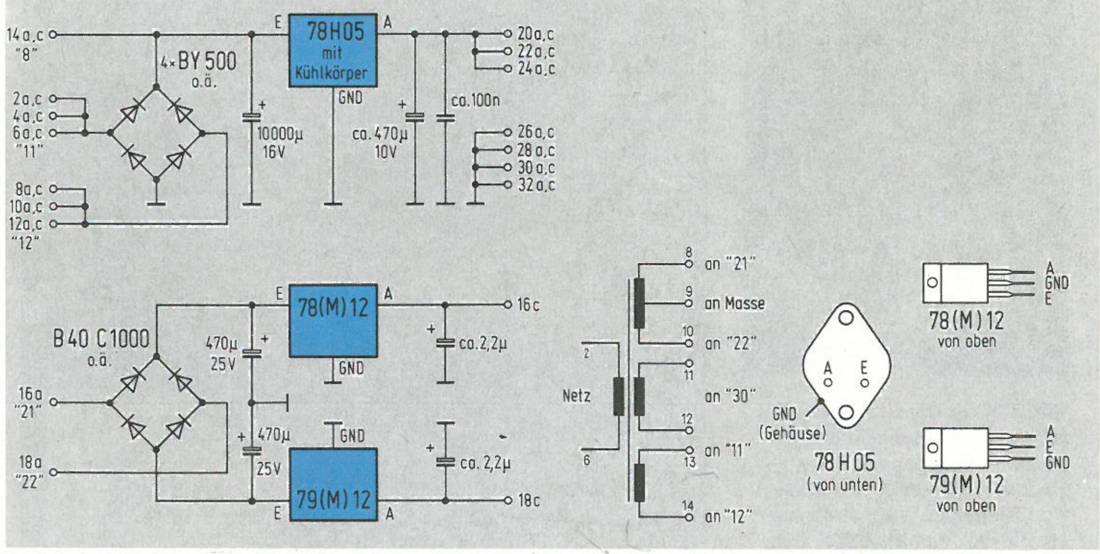


Bild 29: Drei Spannungen liefert die große Netzteil-Karte, die mit Längsreglern aufgebaut ist.

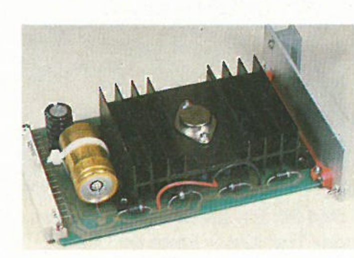


Bild 30: Der Kühlkörper ist so anzuordnen, daß die hindurchführenden Anschlußstifte des TO-3-Reglers Platz für den darunterliegenden Elko lassen.

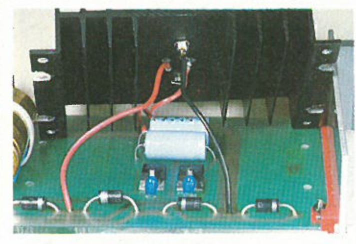


Bild 31: Der Festspannungsregler wird über drei Drähte mit der Platine verbunden.

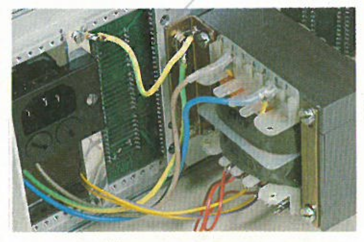


Bild 32: Über Distanzstücke und auf einem eigenen Halteblech ruhend läßt sich der Netztrafo von hinten an die Z-Schienen anflanschen.



Bild 33: Mit einem einzigen Griff lassen sich an dieser Verteilerdose mehrere Geräte gleichzeitig ein- und ausschalten.

Den Speicher auf die Bank verbannt

Um den Hardware-Speicherbereich zu vergrößern, können verschiedene dieser RAM/ROM-Karten parallel geschaltet werden. Damit man jede gezielt ansprechen kann, müssen weitere Adreßbits herangezogen werden, die die CPU in Form der „künstlichen" Adreßbits A16...A18 generiert (Banking-Adreßverfahren)! Entsprechend muß auf jeder Speicherkarte eine andere Bank-Adresse 0...7 eingestellt werden. Mit einem in der Adreßleitung A15 (oberstes Adreßbit) liegenden Inverter kann man zusätzlich festlegen, ob die 32 K RAM in der oberen (kein Inverter) oder unteren Hälfte (mit Inverter) des regulären 64-K-Adreßraums liegen sollen. Da für diesen Inverter kein IC mehr unterzubringen war (das ohnehin nur spärlich ausgenutzt wäre), läßt er sich (bei Bedarf) in Form eines in die Ecke gequetschten Transistors nachrüsten (**Bild 38**). Wer will, kann statt der vorgesehenen 2732-EPROMs auch die kleineren 2716-Typen einsetzen; an den **Anfangsadressen** ändert sich dadurch nichts, nur ist „hinter" jedem 2716 ein 2-K-Loch, das frei bleibt. Beim Einsatz von 2716-EPROMs ist die Brücke „16" einzulöten, nachdem die Brücke „32" (auf der Platinenunterseite) aufgetrennt worden ist. Diese zusätzlich gebotenen Leistungsmerkmale interessieren im Augenblick wohlgemerkt noch nicht; Spezis unter Ihnen registrieren aber hier bereits mit einem Augenzwinkern die Weitsichtigkeit bei der Konzeption.

Wichtiger Hinweis: Sie brauchen keine der Brücken zu bestücken, wenn die Karte in einem ganz normalen 64-K-System eingesetzt wird, weil auf der Unterseite der Platine die hierfür erforderlichen Brücken bereits vorgesehen sind. Bei Parallelschaltung mehrerer Speicherkarten müssen diese Brücken getrennt werden, ehe Sie andere verdrahten!
Zur Bestückung dieser Karte gehört wiederum ein waches Auge, um einen sich einschleichenden Fehler sofort lokalisieren und beheben zu können. Gehen Sie darum bitte genau nach dem angegebenen Schema vor und stoppen Sie jedes weitere Einlöten, bis ein auftretender Fehler behoben ist! Löten Sie zunächst die TTL-ICs, die Tantalperlen und die VG-Leiste ein sowie die 24poligen Fassungen für die RAMs (also beginnend bei Platz 80; vgl. Bild 36). Danach setzen Sie zunächst erst eins und dann nach und nach immer nur **ein** weiteres RAM ein, aber erst, wenn nach jedem Einsetzen der nachfolgend beschriebene Speichertest geklappt hat (dazu ist der rote Monitor erforderlich, der bei dieser Ausbaustufe aber ohnehin vorhanden sein dürfte).

Der rote Monitor als Prüfhilfe

Sie laden ab 2800 in Ihren Arbeitsspeicher die Sequenz

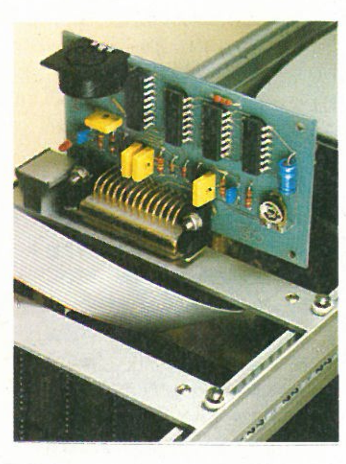


Bild 34: Das Cassetten-Interface kann man über vier Drahtbrücken mit der seriellen Schnittstelle der CPU verbinden; eleganter (aber auch teurer) ist der Anschluß über den 25poligen Subminiatur-Steckverbinder.

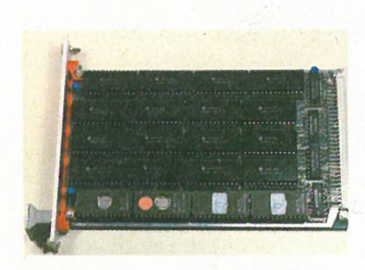


Bild 35: Insgesamt kann die große Speicherkarte 262 144 Bits an RAM/EPROM beherbergen.

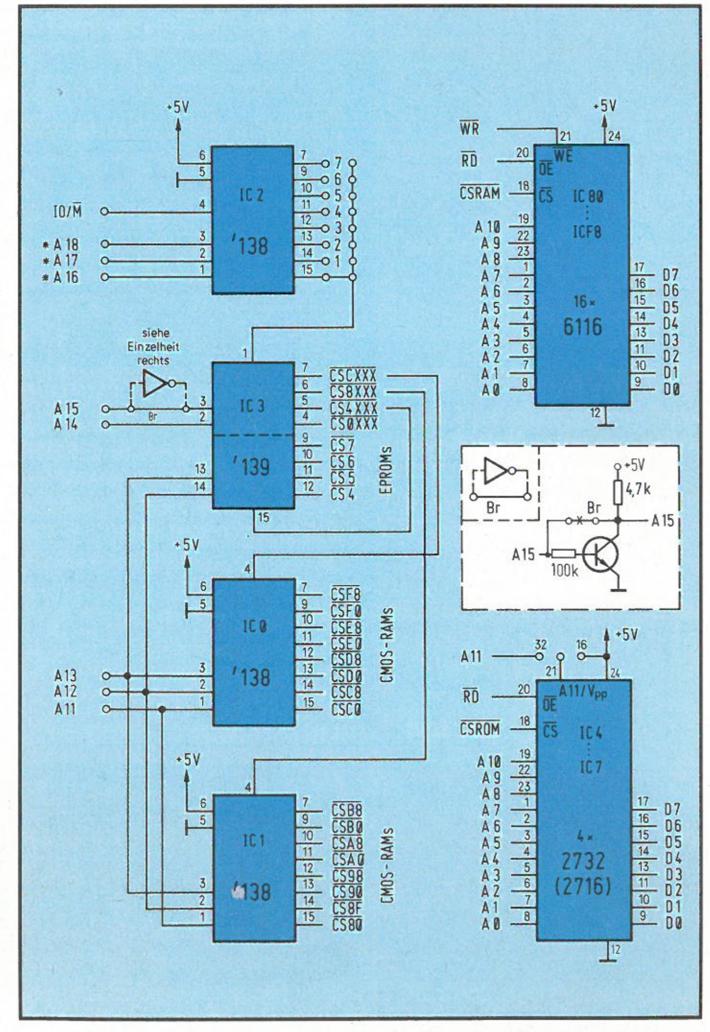


Bild 37: Bestückungsschema der 48-K-Speicherkarte; für den Standardeinsatz braucht keine der Brücken bestückt zu werden.

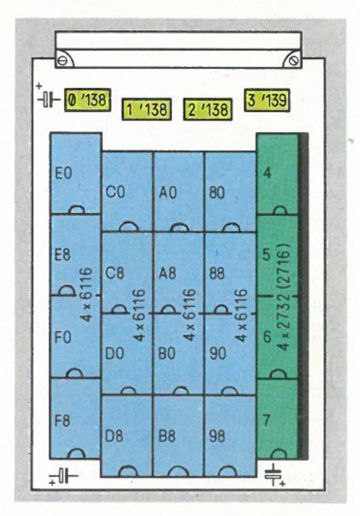


Bild 36: Dank der 2-K- bzw. 4-K-Organisation der verwendeten Speicherbausteine gestaltet sich die Erzeugung der 20 Chip-Select-Signale besonders einfach.

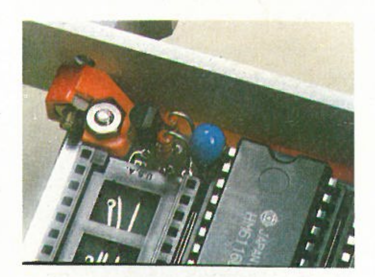


Bild 38: Ein als Inverter arbeitender Transistor kann den Adreßbereich des 32-K-RAM-Blocks auf die untere Hälfte des regulären 64-K-Adreßraums verschieben (nur bei Systemerweiterungen).

nach **Bild 39** und starten diese bei eingesetzter Speicherkarte (die Absolutadressen für die angegebenen symbolischen Namen finden Sie auf der Symboltabelle, die Ihrer Monitor-Kurzbeschreibung beiliegt); nach kurzer Zeit erscheint in der HEX-Anzeige die höchste Speicheradresse, die das Programm für in Ordnung befunden hat; im Normalfall ist es die höchste Adresse des gerade eingesetzten Speicherblocks, also 87FF beim ersten RAM. Ist dies der Fall, können Sie das nächste RAM einsetzen und testen und so fort, bis Sie beim letzten angelangt sind. Mit dieser Testroutine (die Ihr BASIC später ständig aufruft) können Sie sicher sein, daß von Adresse 8000 an bis zur angezeigten Adresse sämtliche RAM-Zellen lückenlos in Ordnung sind (darum muß die Bestückung auch von unten nach oben, und zwar lückenlos, erfolgen).
Erscheint in der Anzeige eine andere als die zu erwartende Adresse, so ist das RAM (zumindest teilweise) defekt; erscheint gar nichts mehr, haben Sie beim Löten einen Fehler gemacht, den Sie schnellstens suchen und beheben sollten! Bei „7FFF“ in der Anzeige ist das RAM auf der Speicherkarte gar nicht angesprochen worden (was auch passiert, wenn die Karte nicht eingesteckt ist!).

Achtung: Bei voll bestücktem Speicher dauert dieser Test gut zwei Sekunden; gegenüber den bisher vernachlässigbar kurzen Laufzeiten fällt dies bereits deutlich ins Gewicht, was Sie bitte berücksichtigen, wenn Sie derartige Programmteile in eigene Programme einbeziehen.
Sie sollen nun noch den Hintergrund dieses kurzen Programmstücks kennenlernen, dessen Sie sich hier bedienen: Das Unterprogramm MEMCHK füllt, beginnend bei 8000 (der ersten RAM-Adresse auf der großen Speicherkarte), den gesamten Adreßraum bis FFFF (der höchsten Adresse überhaupt) mit einer Konstanten; dabei schert sich das Programm nicht darum, ob die RAMs auch tatsächlich bestückt sind, denn das bekommt es bei der anschließenden Abfrage heraus. Es testet nämlich im zweiten Durchgang, welches die letzte Adresse ist, unter der es die Konstante wiederfindet. Da dieser Test sequentiell mit allen RAM-Zellen passiert, kann man davon ausgehen, daß im getesteten Bereich alle in Ordnung sind. Beim Rücksprung aus MEMCHK steht diese höchste Adresse übrigens in den Registern A und B (obere Hälfte in A).

```
2800  CD 5B 00  TEST   CALL MEMCHK  Monitor-Sub aufrufen
  03  2A EO 2F         LHLD 2FEO    oberste Adresse holen
  06  22 CD 2F         SHLD ADRLO   ins Adreßfeld laden
  09  CD yy xx  TELOP  CALL READ    Ergebnis anzeigen
  0C  C3 09 28         JMP TELOP    Endlosschleife
```

Bild 39: Das im roten Monitor enthaltene Unterprogramm MEMCHK dient zum Speichertest, den Sie mit Hilfe dieser Sequenz durchführen können (xx yy: READ-Adresse siehe Symboltabelle S rot).

MOPPEL mausert sich

Mit den unterschiedlichen Monitor-Versionen haben Sie effektive Testhilfen an die Hand bekommen. Gewissermaßen als Krönung können Sie eigene Programme (oder Daten, wie z. B. Geburtstage o. ä.) fest in EPROMs ablegen und sie auch mit Hilfe des kleinen Druckers dokumentieren. Die Ansteuerung dieser scheinbar simplen Baugruppen erfordert bereits einen ganz beachtlichen Software-Aufwand. Die entsprechenden Verwaltungsprogramme brauchen Sie aber nicht etwa selbst zu erstellen; sie sind fest im EPROM untergebracht (im Monitor bzw. im orangen Drucker-EPROM), und Sie müssen sie nur geeignet aufrufen.

Wem die Stunde schlägt

Für einen ordentlichen Computer gehört es sich, daß er stets die Uhrzeit und das Datum bereithält, und zwar ohne daß man ihm nach dem Einschalten erst einmal „sagen“ muß, was die Stunde geschlagen hat. Sie werden an den späteren Programmbeispielen sehen, wie unerläßlich diese Zeitinformation ist, und mit Hilfe der kleinen Echtzeit-Uhr können Sie MOPPEL mit dieser Zusatzfunktion ausrüsten (**Bild 40**). Das Herzstück dieser Baugruppe (die nicht ohne Grund so abgemagert ist) bildet der CMOS-Uhrenschaltkreis MSM5832 (vgl. „Das interessante IC“ in ELO 8/82), der bereits für den Anschluß an ein Mikrocomputer-System vorbereitet ist (**Bild 41**). Das Einschreiben (Stellen) und Auslesen der Uhr erfolgt digitseriell auf vier Datenleitungen, d. h. nacheinander werden Sekunden-Einer, Sekunden-Zehner... bis Jahres-Zehner geladen bzw. gelesen. Im IC nehmen 13 einzelne 4-Bit-Register die Zeitinformation auf, und ein Quarzoszillator sorgt in Verbindung mit einer Zähllogik dafür, daß Uhr und Kalender wohlgeordnet weiterlaufen. Zur Adressierung dieser Register dienen vier Adreßbits, zu denen sich weitere vier Steuerleitungen (RD, WR, HOLD und ADJ) gesellen.
Aus einem naheliegenden Grund läßt sich dieses IC nicht so einfach ansprechen wie etwa ein RAM; denn wenn Sie die Uhrzeit just in dem Augenblick auslesen, wo ein signifikanter Sprung stattfindet (z. B. von 19:59:59 h auf 20:00:00 h), könnten Sie zu Beginn des Lesens die „alten“ Stunden und am Ende die „neuen“ Minuten erwischen, weil der Lesevorgang selbst ja auch seine Zeit in Anspruch nimmt. Heraus käme dann 19:00:00 h statt 20:00:00 h, und das ist sicherlich nicht im Sinne einer Computeruhr! Folglich muß man das Lesen mit einem „Anklopfen“ beginnen, indem man die HOLD-Leitung auf HIGH bringt; das IC weiß dann, daß es interne Umschaufelvorgänge abschließen (oder neue ein wenig verzögern) muß, weil man höheren Ortes das Auslesen begehrt. Und das IC richtet sich danach; 150 µs nach der Ankündigung steht die Zeit stabil und kann ausgelesen werden. Wegen dieser etwas komplizierten Zeitabläufe

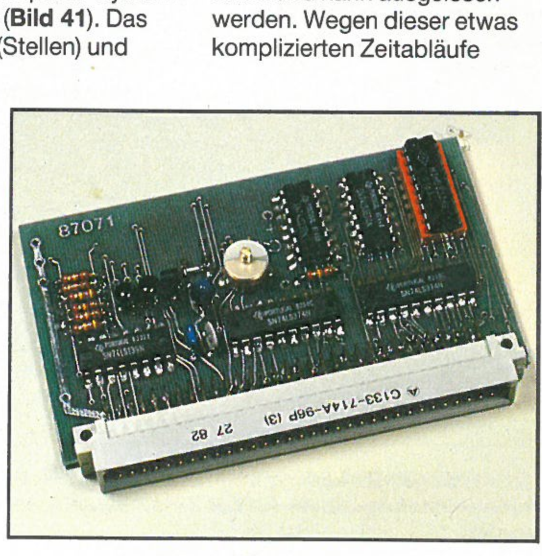


Bild 40: Ein CMOS-Uhren-IC auf der Platine der Echtzeit-Uhr übernimmt das zeitrichtige Weiterzählen von Uhrzeit und Datum.

Bild 41: Wegen der komplizierten Zeitabläufe beim Einschreiben und Auslesen muß das Uhren-IC über Zwischenpuffer angesteuert werden.

müssen sämtliche Leitungen zum Mikrocomputer zwischengepuffert werden, was die beiden Speicher-ICs 74374 übernehmen.

Umschalten auf Sparflamme

Das IC erhält bei eingeschalteter System-Stromversorgung seine +5 V über den leitenden Transistor T1; gleichzeitig wird hierüber (und den 1-kΩ-Widerstand) der Puffer-Akku geladen, der die Gangreserve bildet. Geht nämlich nach dem Abschalten die Systemversorgung weg, liefert der Akku über den Vorwiderstand genügend Saft, um den Gang des Uhren-ICs sicherzustellen. Dieser Akku darf im System übrigens nur einmal bestückt sein, also entweder auf der CPU oder hier auf der Echtzeit-Uhr; in jedem Fall wird der andere zu puffernde Partner (das CMOS-RAM oder das CMOS-Uhren-IC) über die Leitung 24a (UvCMOS) mitversorgt. Ein weiteres Detail haben Sie mit wachem Blick erkannt und harren auf Erklärung: Da sitzt noch ein unscheinbarer Transistor T3, dessen Kollektor vielsagend vor drei Brücken herumbammelt. Die drei Brücken führen an die Hardware-Interrupt-Eingänge des 8085 (RST5.5, RST6.5 und RST7.5). Bei entsprechender Vorbereitung (und Verdrahtung höchstens *einer* Brücke) läßt sich so im Sekundentakt ein Interrupt auslösen, um beispielsweise Uhrzeit und Datum auf dem Bildschirm zu aktualisieren. Das ist im Augenblick noch zu früh für uns, und wir lassen diesen Schaltungsteil deshalb noch in Frieden. Wie bei einer guten Aussteuer können Sie aber auch hier das Gefühl haben, daß das Ganze auf Zukunft ausgelegt ist.

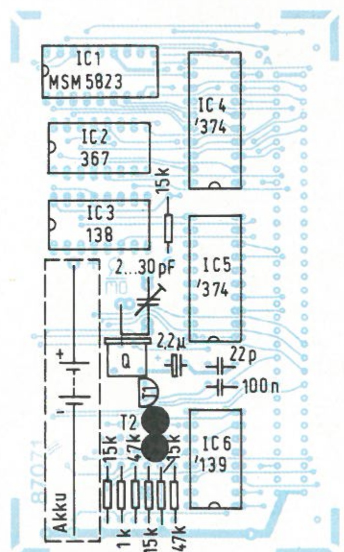


Bild 42: Nur das Uhren-IC soll beim Bestücken auf eine Fassung gesetzt werden; der Akku darf nur einmal im System vorhanden sein, also entweder auf der CPU oder hier auf der Echtzeit-Uhr.

Ran ans Werk

Das richtige Bestücken dieser Winzlings-Platine ermöglicht Ihnen **Bild 42**. Dabei sollte nur das Uhren-IC auf eine Fassung kommen, die TTLs sind besser aufgehoben, wenn sie eingelötet werden. Vielleicht kommt Ihnen der verwendete Quarz etwas mickerig vor (**Bild 43**), aber damit hat es

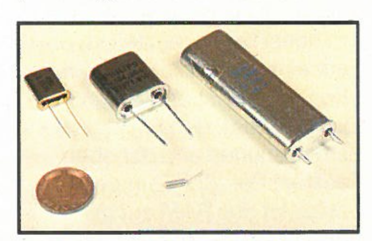


Bild 43: So verschieden können Quarze aussehen: Im Vergleich zu seinen größeren Brüdern ist der Subminiaturquarz kaum wiederzufinden.

vollauf seine Richtigkeit. Es ist ein Typ, den Sie auch in Ihrer *Quartz*-Armbanduhr wiederfinden, und dementsprechend winzig ist er. Behandeln Sie ihn mit Vorsicht, er fliegt leicht mit Lötzinn-Resten in den Abfall! Der auf dem zarten Quarzgehäuse festzulötende Bügel ist, wie bei jedem Quarz, obligat. Den Oszillator gleichen Sie mit einem Zähler ab, den Sie am Testpunkt neben dem Trimmer (**Bild 44**) und an Masse anschließen. Dann laden und starten Sie bitte folgende Sequenz: MVI A,F4; OUT 19; HLT (3E – F4, D3 – 19, 76; RES, RUN). Damit stellen Sie für das Uhren-IC den Ruhezu-

stand her, in dem es an den Testpin eine Frequenz von 1024 Hz liefert; der Abgleich läßt sich besser (weil schneller) über die Periodendauer durchführen, die Sie auf möglichst genau 976,5625 µs einstellen sollten.

Ticken im Verborgenen

Es kommt der Tag, da Sie die Uhr stellen wollen; dazu bedienen Sie sich der Funktion **WRITUR** (FCT 8 im roten Monitor) unter der Berücksichtigung, daß für den Wochentag Montag, Dienstag bis Sonntag jeweils 01, 02 bis 07 einzusetzen ist. Sollte es, wie beim Abfassen dieser Zeilen, gerade 10 Uhr 47 sein und dazu Sonntag (= 07), der dritte Oktober 1982, müßten Sie folgendes eingeben: FCT – 8, 1047 (für 10 Uhr 47) NXT, 0703 (für Sonntag, den dritten) NXT, 1082 (für Oktober 1982), gefolgt von RUN. Der Monitor zerlegt diesen Zahlenwirrwarr in appetitliche Häppchen, die er der Echtzeit-Uhr überreicht; zu Ihrer Kontrolle erscheint die Uhrzeit unmittelbar danach in der HEX-Anzeige, was Ihnen mit dem sanften Weiterticken die Gewißheit gibt, daß die Uhr richtig„geht".
Achtung: Das Weiterlaufen der Uhrzeit erfolgt mit dem Demonstrationsprogramm DIGUR, das nur Software-Zeitverzögerungen (und keinen Quarztakt) verwendet; daher geht die angezeigte Uhrzeit nach einer Weile falsch, während die Echtzeit-Uhr intern richtig weiterläuft (wovon Sie sich durch erneutes Auslesen überzeugen können)!
Zum Auslesen der Uhrzeit geben Sie ein „FCT – 9" (im roten Monitor), gefolgt von RUN; sofort holt Ihnen der Monitor daraufhin die aktuelle Uhrzeit in die HEX-Anzeige und läßt sie (wiederum über das Demo-Programm DIGUR) weiterzählen. Wenn Sie außer FCT – 9 noch die Adresse eines freien RAM-Bereichs eingeben, bevor Sie per RUN loslegen, überschreibt der Monitor die Uhrzeit-/Datums-Information (im BCD-Format) in diesen RAM-Bereich. Aber da ist noch ein weiteres Zeit-Unterprogramm mit dem dezenten Namen „TI" (Time Input; 0061). Das hat die entzückende Eigenschaft, Uhrzeit und Datum nicht nur aus der Echtzeit-Uhr auszulesen, sondern alles auch noch in den ASCII-Code umzusetzen und in einen RAM-Hilfspuffer zu überschreiben. So etwas werden Sie schon sehr schnell zu schätzen wissen; diese Umsetzung in den ASCII-Code (das spricht man „Aski", und um Himmels willen nicht „A-Es-Zeh-Zwei"!) bereitet die Informationen Uhrzeit und Datum nämlich derart auf, daß sie von anderen Peripheriegeräten verstanden und verarbeitet werden können. Konkret: Wenn Sie Ihren Thermodrucker (der auch nur „ASCII" versteht) mit so einer Information füttern, druckt der die so aus, wie es sich ein vernünftiger Mensch vorstellt: So,03.10.82; 10:47:23h (vgl. auch Bild 19). Das Buchstaben-Kürzel für den Wochentag erzeugt der Monitor übrigens ohne viel Aufhebens selbsttätig; denn wer will schon hören (oder lesen), daß heute „04" ist – da klingt „Donnerstag" doch eindeutig wohliger.

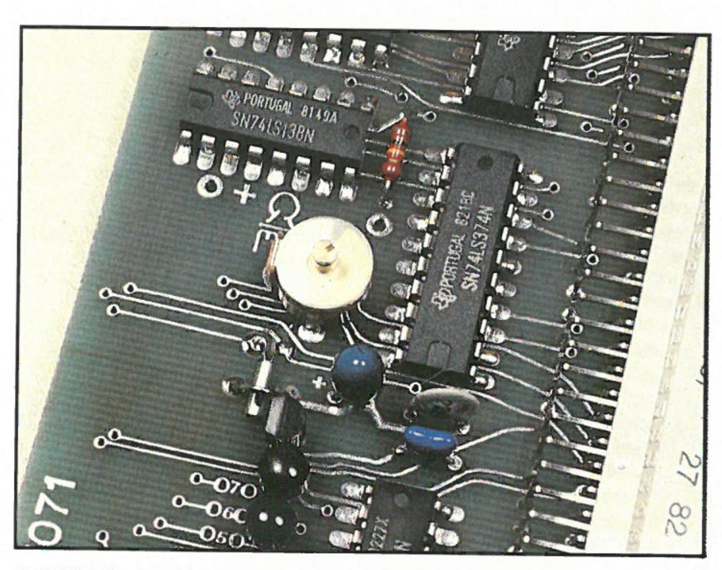


Bild 44: Mit dem Trimmer erfolgt der Abgleich des Oszillators im Uhren-IC; an dem Testpunkt daneben können Sie dazu Ihren Zähler anschließen.

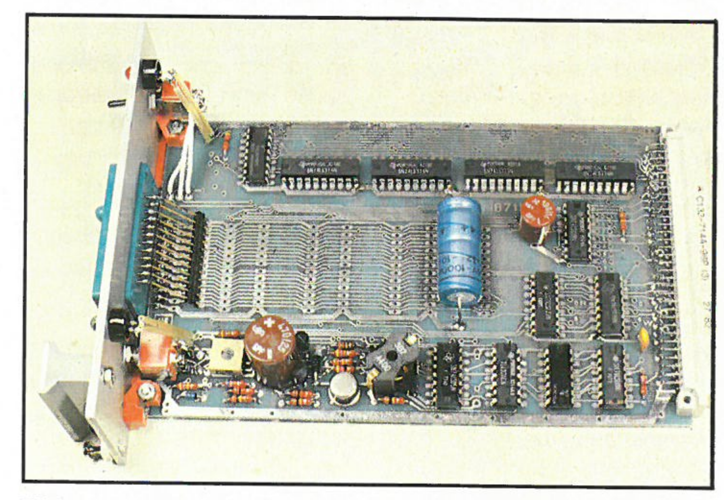


Bild 45: Der EPROM-Programmierzusatz erweitert Ihren MOPPEL zum Programmiergerät für die Standard-Typen des Marktes.

Der MOPPEL-Prommer

Obwohl in der Mikrocomputerei bei der Sprachhandhabung nicht gerade zimperlich umgegangen wird (man rettet *Register* und meint deren Inhalt usf.), sollten Sie eins klar sehen: Im MOPPEL programmieren Sie EPROMs mit einer Erweiterungskarte, einem EPROM-*Programmierzusatz* also (**Bild 45**); das ist kein *Programmiergerät* im eigentlichen Sinne (was es natürlich auch gibt), weil die Karte eben nicht eigenständig lebensfähig ist. – Wer sich an den Nachbau dieser Karte macht, sollte den elektronischen Kinderschuhen entwachsen sein; denn auf dieser Karte passiert etwas, das zur Katastrophe ausarten kann, wenn man nicht diszipliniert vorgeht: Es werden hier, mir nichts, dir nichts, Spannungen von über 20 V erzeugt, die zum EPROM-Programmieren erforderlich sind. Und wer da Schraubendreher, Pinzetten oder ähnliches gezielt fallen läßt, kann natürlich einiges zerschießen, was auch bei CPU und Speicher nicht halt macht...
Das wollen Sie bitte nicht als Abschreckung verstehen, sondern ganz im Gegenteil als wohlgemeinten Rat; denn es gibt nichts Eleganteres, als Programme im System selbst zu entwickeln und anschließend dauerhaft ins EPROM zu überschreiben, was im und mit demselben System passiert, auf dem auch die Programmentwicklung stattgefunden hat. – Lesen Sie hier, quasi zur Vervollkommnung, noch zwei nicht unwesentliche Randinformationen: Der Ausdruck „PROMs brennen" stammt von bipolaren *PROMs* (keine EPROMs!), bei denen die Informationen (irreparabel) eingebracht werden, indem man Halbleiterübergänge definiert zerstört („Sicherungen" gezielt zerschießt); bei EPROMs kann man den Inhalt durch intensive Bestrahlung mit UV-Licht wieder löschen (alle internen Bits auf HIGH setzen), daher das „E" vor dem PROM: *Erasable* (= löschbarer) PROM (= programmierbarer Lese-Speicher). Wie oft man so etwas

machen kann? Die Hersteller sagen, daß es garantiert 1000-mal funktioniert; im Vertrauen darauf, daß das sowieso keiner ausprobiert, haben sie damit sicherlich Recht. Und wie lange hält diese dauerhaft eingebrachte Information? Wiederum garantiert: Zehn Jahre; ob Sie dann noch an das EPROM denken, das Sie heute programmieren, mag dahingestellt bleiben. Wir halten nur fest, daß sich so ein EPROM ganz schön oft programmieren (und löschen) läßt, und daß die Informationen ganz schön lange darin erhalten bleiben!

PROMMER und Drucker für MOPPEL

Beim Blick auf die Schaltung des EPROM-Programmierzusatzes (**Bild 46**) sind vier Blöcke zu unterscheiden: Ganz unten die Decodier-Logik (ICs 6, 11 und 12) für die Baugruppe, die im MOPPEL vier In-/Out-Portadressen belegt ($08...0B); ganz oben ist die Schaltung für die Spannungswandlung angeordnet, die aus der +5-V-Systemversorgung die 25 V (bzw. 21 V) Programmierspannung macht (IC 15 plus rankendes Beiwerk); in IC 5 werden die dazu notwendigen Steuersignale abgelegt. Den dritten Teil bilden die Daten- und Adreßspeicher mit Leselogik (ICs 1...4), die die Verbindung herstellen zwischen Mikrocomputer-Bus und den Programmiersockeln für die EPROMs (ICs 7...10). Obwohl auf der Karte Platz für vier Sockel gelassen wurde, bestücken und behandeln wir nur einen, und das ist der an der Platinen-Schmalseite (Platz #3).
Sie können in den Sockel Standard-EPROMs der Typen 2758 (1 K × 8), 2716 (2 K × 8) sowie 2732 (4 K × 8) einsetzen und diese lesen oder programmieren. Da sich die Stiftbelegung des 2732 von der der kleineren Brüder unterscheidet (die Hersteller haben lange gebraucht, um sich diese Extravaganz einfallen zu lassen!), ist der Umschalter S1 erforderlich, der die Zuführung zu den Pins 20 und 21 vertauschen kann. Der Programmiervorgang selbst spielt sich derart ab, daß man die gewünschte Adresse mit zugehörigen Daten ausgibt, einen 50 ms langen Programmierpuls erzeugt (der pulst die Programmierspannung ans EPROM) und anschließend nachsieht, ob im PROM das steht, was man rein haben wollte (prüflesen). Ist das der Fall, geht es bei der nächsten Adresse weiter, andernfalls meckert der Monitor den Fehler an, indem er „Err" in die HEX-Anzeige bringt und sich weigert, weiterzuprogrammieren; wenn Sie in diesem Fall (nach RES, REG) die Register B und C inspizieren, haben Sie die Adresse, bei der das Programmieren ausgehakt hat. Bei fehlerfreiem Vorgang leuchtet am Ende die grüne LED auf und signalisiert Ihnen, daß das Programmieren ohne Zwischenfälle erfolgreich beendet ist, das jeweilige Prüflesen eingeschlossen.
Zum Programmieren setzen Sie das EPROM (mit der Kerbe nach oben!) in den Sockel ein und verrammeln es durch Niederdrücken des Hebels (**Bild 47**). Dann wählen Sie mit Hilfe des Schalters S1 (unten) den Typ: 2708/16 links bzw. 2732 rechts; dann stellen Sie mit S2 ein, ob Ihr EPROM mit 25 V (Normalfall) oder mit 21 V (2732A) programmiert werden möchte. Diese Einstellerei sollte mit Verstand geschehen, denn während sich ein 2716 (oder ein normales 2732) zur Not auch noch mit 21 V behandeln läßt, nimmt einem ein 2732A das Anlegen von 25 V derart übel, daß es beleidigt kaputtgeht. Darum schreiben die Hersteller schon flehentlich auf diese Exoten drauf, daß man sie tunlichst mit 21 V zu programmieren hat: PRGM @ 21V (**Bild 48**). Warum es diese beiden Möglichkeiten gibt? Böse Zungen behaupten, um den Umsatz zu steigern, weil auf diese Weise reichliche Stückzahlen den Jordan hinabwandern (sprich: das zarte Halbleiterleben aushauchen). Die Hersteller dagegen behaupten, dies sei nun mal die modernere Technologie, und das müssen wir ihnen glauben, zumindest aber müssen wir damit leben!

Auf zu hohen Spannungen

Pfiffige ELO-Leser haben längst bemerkt, daß das Spannungswandler-IC TL497 auch schon in der Rubrik „Das interessante IC" vorgestellt worden ist (Heft 7/82); es arbeitet hier als zerhackender Aufwärtsregler, bei dem eine kleine Induktivität (L1) als Energiespeicher zur Hochtransformation verwendet wird. Diese „Spule" stellen Sie völlig problemlos her, indem Sie 40 Windungen eines Kupferlackdrahtes (ca. 0,3 mm Durchmesser) auf den Kern wickeln und die Enden an den äußeren beiden Lötstiften des Spulenkörpers verlöten (**Bild 49**; jede innere Abneigung gegen so ein Spülchen ist vollkommen unbegründet, das dürfen Sie getrost glauben!). Achten Sie beim Einlöten bitte nur darauf, daß Sie die Spule richtig herum einsetzen und daß die Drahtenden Verbindung mit den Leiterbahnen bekommen (**Bild 50**). Der Abgleich geht dann folgendermaßen vor sich: +5 V anschließen und Pin 2 von IC 15 mit Masse verbinden; nun das Poti derart verdrehen, daß an der Katode der Schottky-Diode genau 21 V zu messen sind. Ohne weiter am Poti zu drehen, schließen Sie S2 und kontrollieren, daß die Spannung auf ca. 25 V ansteigt. Zur Verdrahtung der Schalter brauchen Sie nur so vorzugehen, als ob Sie diese von oben in die Platine einlöten; wenn Sie sie abgewinkelt in die Frontleiste einbauen, müssen die Drähte folglich einen sanften Rechtsknick machen, ohne sich in irgendeiner Weise mit dem Nachbarn zu kreuzen. *Vergewissern Sie sich unbedingt, daß auch tatsächlich diejenige Spannung erzeugt wird, die der Schaltknebel von S2 anzeigt (vgl. Bild 47)!*
Beim Betrieb des Spannungswandlers hören Sie ihn förmlich arbeiten, was von Magnetostriktionseffekten des Schalenkerns herrührt (das Wort sollten Sie auswendig lernen, es schindet Eindruck!); damit sind mechanische Bewegungen (in diesem Fall der beiden Kernhälften) gemeint, die ihre Ursache in den magnetischen Wechselfeldern haben.

Nürnberger Trichter mit Langzeitwirkung

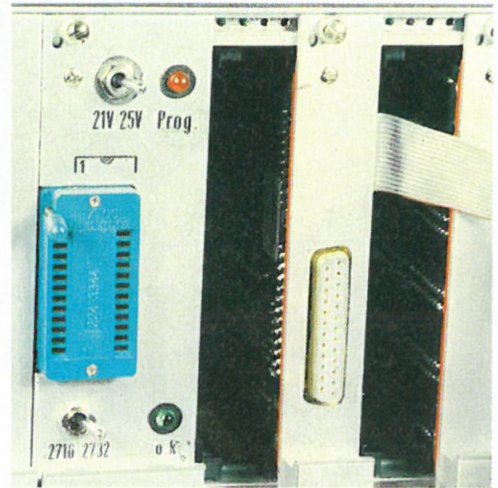
Angenommen, Sie haben im RAM-Bereich 8000...87FF Daten liegen, die Sie gern dauerhaft in einem EPROM wiederfinden möchten, und zwar in einem 2716; dann gehen Sie dazu bitte folgendermaßen vor: EPROM einsetzen und Schalter einstellen wie oben beschrieben; dann FCT – 6 aufrufen (im roten Monitor), gefolgt von der Startadresse des Quellbuffers 8000, NXT; dann Endadresse des Quellbuffers eingeben 87FF, NXT, und die ROM-Anfangsadresse, bei der das Einschreiben beginnen soll (in der Regel 0000, NXT); zum Schluß folgt dann die Angabe des EPROM-Typs „16", RUN. Die Sequenz sollten Sie üben, indem Sie mit Verstand beim Eingeben sagen „Von 8000 (NXT) bis 87FF (NXT) nach 0000 (NXT) in ein

Bild 46: Die Programmierspannung von 25 V (bzw. 21 V) wird über einen Spannungswandler auf der Karte selbst erzeugt.

(27)16, RUN". Den Rest besorgt der Monitor, der zunächst einmal die Programmierspannung einschaltet (die rote LED leuchtet), eine Sekunde lang wartet (was sehr wirkungsvoll aussieht), um dann mit dem eigentlichen Programmiervorgang zu beginnen. Seine Arbeit können Sie akustisch (beim piepsenden Spannungswandler) sowie optisch in der LED-Zeile der HEX-Tastatur verfolgen: Dorthin werden nämlich die obersten acht Adreßbits derjenigen Adresse übertragen, die beim PROM gerade am Wickel ist (von einer Adresse „3E7" sehen Sie demzufolge nur die „3E" in der LED-Zeile). Am zyklischen Weiterschalten dieser Adresse können Sie das schrittweise Vollprogrammieren verfolgen, was, wie oben beschrieben, entweder mit der strahlend grünen Erfolgs-LED oder der Err-Fehleranzeige beendet wird.

Das Lesen eines EPROMs vollzieht sich so ähnlich per FCT – 7 (im roten Monitor): Von PROM-Adresse 0000 (NXT) bis PROM-Adresse 7FF (NXT) nach RAM-Adresse 8000 aus einem (27)16 (bzw. 32), RUN. Jetzt leuchtet die grüne LED eine Sekunde lang auf, um den Einlesevorgang anzuzeigen; der geht in Wirklichkeit natürlich hundertmal schneller vor sich, was aber nach nichts aussieht, daher diese künstliche Verlängerung (so viel Zeit muß sein!). Natürlich können Sie mit Hilfe dieser

Bild 47: Die gravierte Frontleiste verleiht dem Ganzen ein gefälliges Aussehen.



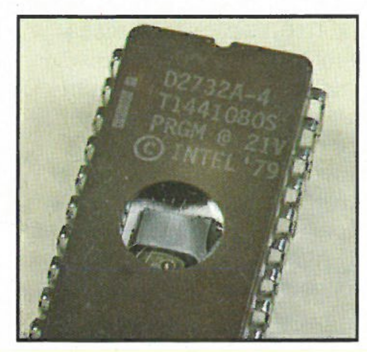


Bild 48: Vorsicht bei den A-Typen des 2732: Sie dürfen nur mit 21 V programmiert werden, wie es die Aufschrift auch angibt.

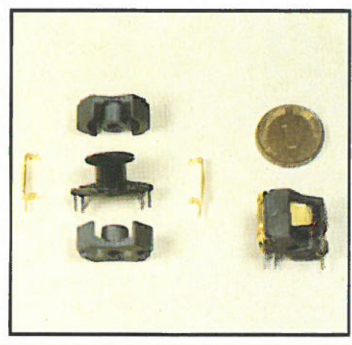


Bild 49: Nach dem Bewickeln des Spulenkörpers werden die beiden Kernhälften mit den Klammern zusammengehalten.

beiden Funktionen auch EPROMs (oder Teile davon) duplizieren, indem Sie erst eins auslesen und dann ein anderes programmieren; selbstverständlich kann das bei beliebigen, auch von null verschiedenen Startadressen aus beginnen (sowohl lesen als auch programmieren), so daß man ein EPROM auch abschnittweise füllen oder Inhalte verschieben kann.

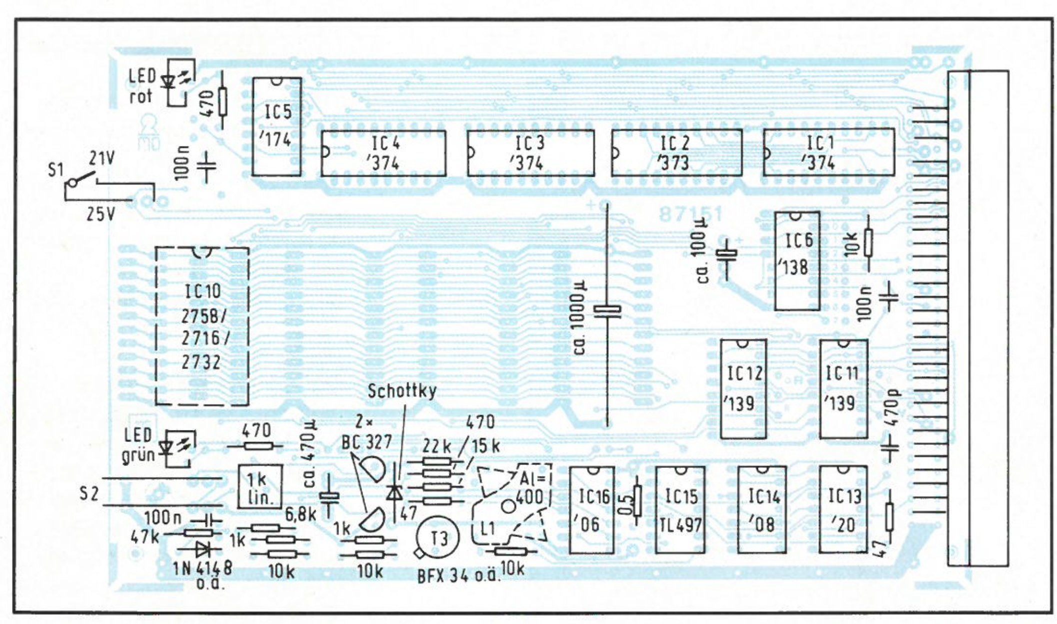
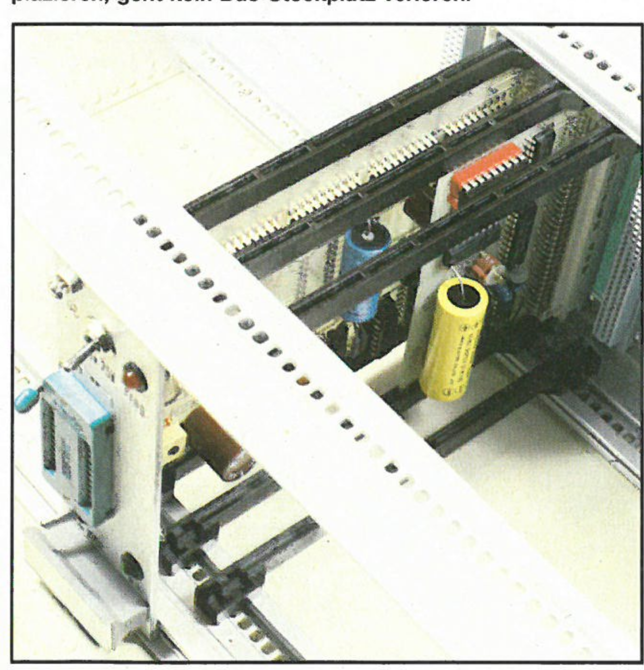


Bild 50: Je nach Anwendungsfall läßt sich der Programmiersockel direkt in die Platine einlöten oder abgewinkelt über Wrap-Pfosten montieren.

Bild 51: Wenn Sie die Echtzeit-Uhr neben dem Prommer plazieren, geht kein Bus-Steckplatz verloren.



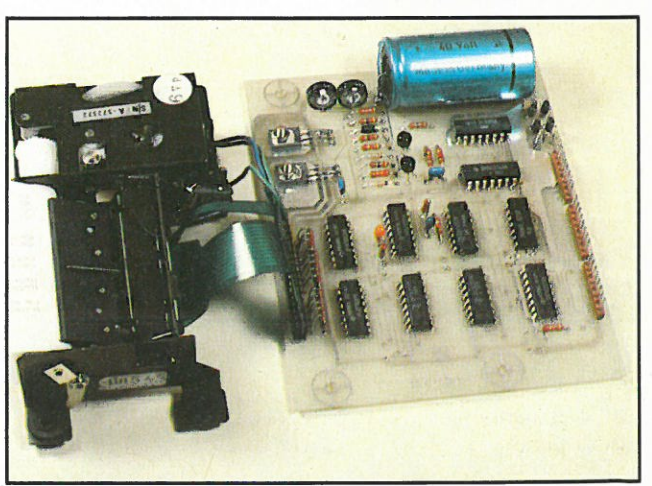
Eine Platinen-Ehe

Wenn Sie versehentlich statt des Auslesens (FCT – 7) das Programmieren (FCT – 6) aufrufen, zerschießen Sie sich das eingesetzte EPROM; in dem 1-s-Vorspann, in dem sich die Programmierspannung aufbaut, können Sie aber noch RESET drücken und damit verhindern, daß etwas Nachteiliges passiert.
Wegen der Überbreite des Programmiersockels nimmt die Programmierkarte die Breite von zwei Bus-Plätzen ein; dennoch verschenken Sie hiermit nichts, weil die Echtzeit-Uhr den freien Platz neben dem Prommer ausfüllen kann (**Bild 51**); und nun wissen Sie auch, warum diese Platine so abgemagert worden ist.

Der Thermodrucker brennt wirklich ein

Wenn Sie schon keine EPROMs „brennen", wie Sie oben erfahren haben, dann kommen Sie mit dem kleinen Thermodrucker doch noch auf Ihre Kosten, denn der brennt die Informationen (wenn auch sehr zart) wirklich ins Papier ein (**Bild 52**). Zur Versorgung dieser Baugruppe ist das im vorigen Teil beschriebene große Netzteil mit dem speziellen Trafo erforderlich, um die Oberspannung für die Thermoköpfe zu erzeugen. Diese 18-V-Wechselspannung gelangt an eine Gleichrichter-

Bild 52: Nach einem komplizierten Verfahren setzt das Druckwerk die einzelnen Zeichen aus kleinen Pünktchen zusammen.



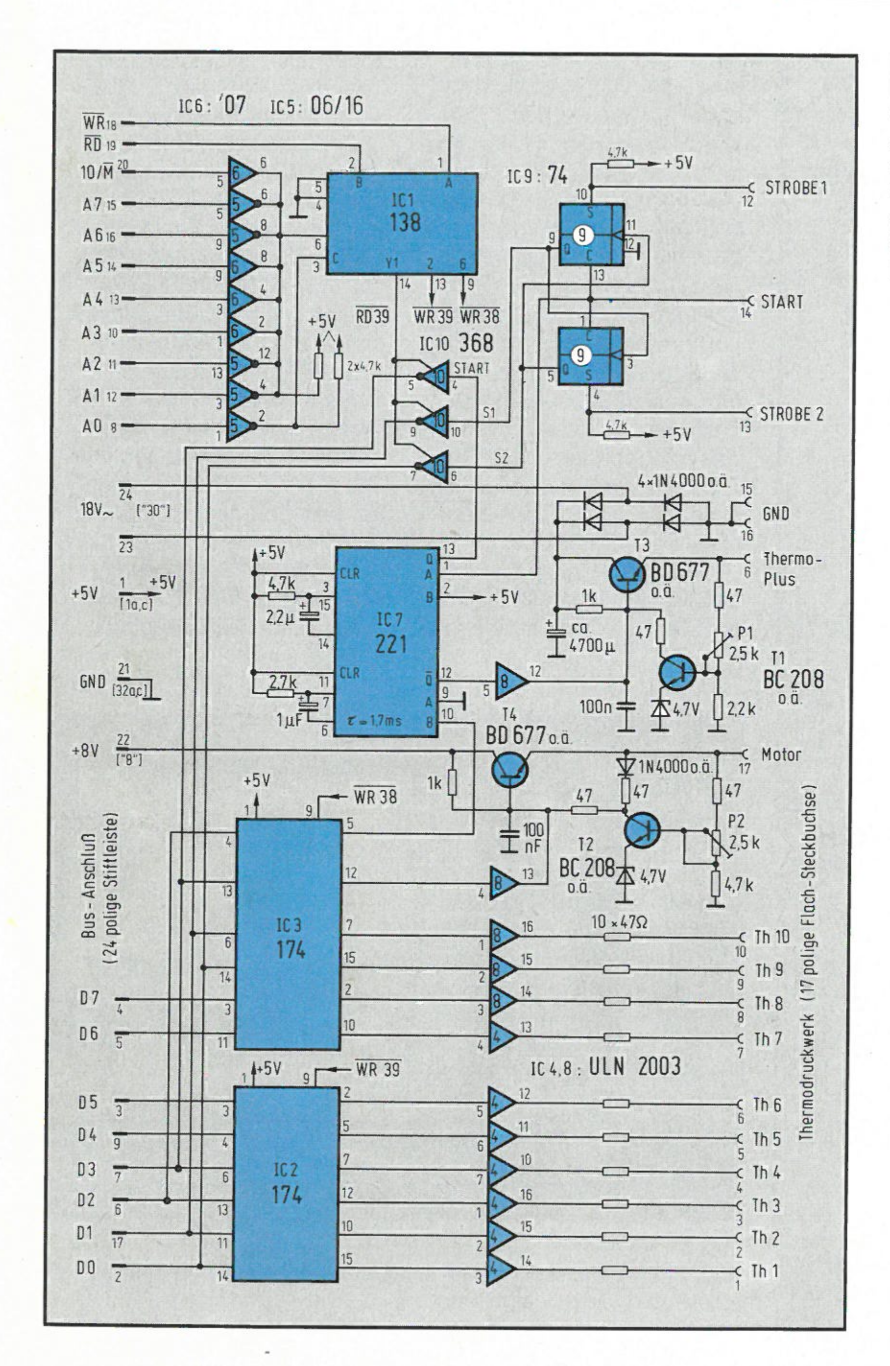


Bild 53: Bei der Drucker-Elektronik werden zwei systemfremde Spannungen verarbeitet: +8 V für den Motor und ca. 24 V für die Thermoköpfe.

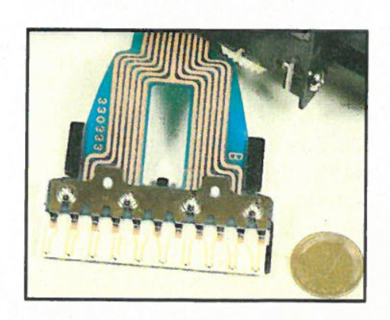
brücke, an die der Schalttransistor T3 angeschlossen ist (**Bild 53**). Dieser Transistor wird (über den Treiber in IC 8) von einer monostabilen Kippstufe (in IC 7) angesteuert, und da haben Sie auch schon die kritische Stelle dieser Schaltung: Wenn T3 zu lange an ist, brennen die winzigen Thermoköpfe unweigerlich durch. Damit dies nie passieren kann, erfolgt die Ansteuerung auf diesem Umweg; lamentieren Sie bitte nicht wegen dieses Verhaltens, denn aufpassen muß man überall, und zu Recht heißt das Motto: Gefahr erkannt – Gefahr gebannt!

Darum gilt auch hier wieder: Vor dem Löten *lesen*! Nachdem wir in der ELO bereits einen Normalpapierdrucker vorgestellt hatten, sind wir nur deshalb beim Thermodruckwerk gelandet, weil der Preis für das andere mittlerweile in astronomische Höhen geklettert ist (Yen-Basis); und mit diesem Aufbau hier ist es heutzutage immer noch möglich, einen Thermodrucker zum (Bausatz-)Preis von DM 299,– anzubieten. Als besonderes Bonbon kann der nicht nur Groß- und Kleinschreibung, sondern Sie können sich einen eigenen Zeichenvorrat generieren, der bis hin zu Grafik-Symbolen reicht. Nun sind Sie wieder versöhnt, oder?

Scheibenwaschen bitte auch

Um die Schaltung weiter zu verstehen, ohne dabei aber bis ins letzte Detail vorzudringen, sollen Sie erfahren, wie unser Thermodrucker zu Werke geht. Da sind, friedlich nebeneinander liegend, zehn winzige Thermoköpfe angeordnet, die sich beim Drucken um die Breite von zwei Zeichen auf dem Papier hin- und herbewegen (**Bild 54**). Stellen Sie sich dazu einmal eine Hausfrau (oder einen Tankwart) beim Scheibenputzen vor; das beginnt links oben und geht bahnweise hin und her nach rechts unten. Von denen hat's der Thermodrukker gelernt, denn er macht es genauso. Nur drückt er den Lappen nicht ständig auf, sondern tippt ihn bei seiner Bewegung nur hin und wieder aufs Papier, so daß dabei kleine Pünktchen entstehen; diese ergeben, wenn sie wohlsortiert auftreten, dann den gewünschten Buchstaben oder ein Sonderzeichen. Um dieses Sortieren brauchen Sie sich aber keine Sorgen zu machen, das nimmt Ihnen der MOPPEL ab; dazu setzen Sie nur (das im Bausatz enthaltene) EPROM auf Platz # 2 der CPU ein (Brücke 3 einlöten!).
Bei der Bestückung nach **Bild 55** lassen Sie bitte zunächst alle ICs weg und löten nur die analogen Bauteile um T1...T4 ein; die Leistungstransistoren T3 und T4 kommen mit dem Gesicht auf die Platine! **Die zehn Vorwiderstände von 47 Ω zu den Thermoköpfen bleiben zunächst noch fern der Leiterplatte!** Die Verbindung zur Busplatine stellen Sie über die mitgelieferten Flachkabel her. Auf jeden Fall muß die Verbindung zur Busplatine hergestellt sein und der Trafo (wie im vorigen Teil beschrieben) verdrahtet werden, um die Drucker-Platine in Betrieb nehmen zu können. Stellen Sie mittels Poti P1 am Emitter von T3 ca. 20 V

Bild 54: Zehn kleine Thermoköpfe brennen bei ihrer Hin- und Herbewegung das Punktraster in die Papieroberfläche ein.



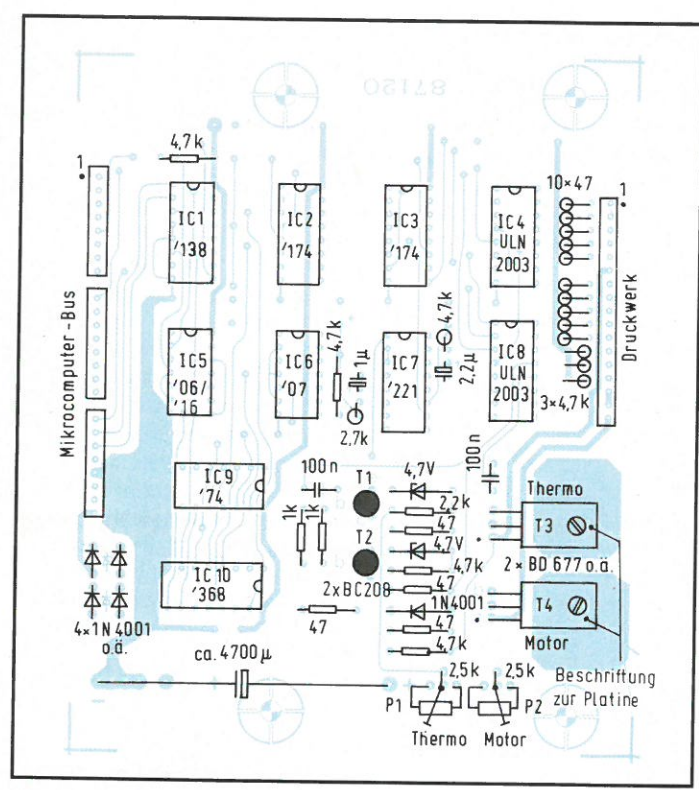


Bild 55: Beim Bestücken ist zu beachten, daß die Leistungstransistoren mit der Schriftseite zur Platine eingelötet werden müssen.

ein, und am Emitter T4 mittels P2 etwa 6 V. Dann klemmen Sie alle „Hochspannung" führenden Teile ab, legen Sie außer Reichweite und bestücken den Rest *außer den zehn 47-Ω-Widerständen*. Nun kommt ein erster Test der Schaltung, zu dem Sie folgende Sequenz nach 2800ff. laden und starten: MVI A, 04; OUT 38; MVI A, 00; OUT 38; JMP 2800. Damit wird das Monoflop IC 7 pausenlos getriggert, was Sie an den Pulsen am Ausgang 5 verfolgen können; treten hier HIGH-Pulse auf, ist das schon ein gutes Zeichen. Wenn Sie zusätzlich über ein Oszilloskop verfügen, sollten Sie die Länge dieser Pulse kontrollieren; sie dürfen nicht länger als 1,7 ms auf HIGH gehen, um die Thermoköpfe nicht zu gefährden; mit dem RC-Glied an den Pins 6 und 7 von IC 7 können Sie (bei Bedarf) diese Zeit variieren (Vergrößern von R oder C verlängert die Mono-Zeitkonstante und umgekehrt). Das sind, wohlgemerkt, Feinheiten für Leute, die bis ins Ultraletzte vorstoßen wollen; im Normalfall genügt es, das Auftreten von *Pulsen* zu

```
2000   C3 89 21 C3
       73 20 C3 36
2008   20 C3 51 20
       00 00 C3 31
2010   20 21 1D 20
       3E 01 32 D3

Gratulation,
er geht
```

Bild 56: Die im Drucker-EPROM enthaltenen Programme können wahlweise einen Speicherabzug anfertigen (HEX-Listing) oder auch Klartexte ausdrucken (ASCII-Zeichen).

kontrollieren (Dauer-HIGH-Pegel wäre der Tod für die zarten Thermoköpfe).

Glückwunsch an Sie: Ich bin o. k.

Nach Abschluß dieses Tests können Sie mit einem befreienden Gefühl auch die restlichen zehn 47-Ω-Widerstände einlöten und das Druckwerk anschließen. Das geschieht durch vorsichtiges Einführen des 17poligen Flex-Prints in die zugehörige Buchsenleiste. Wenn Sie nun noch das passende Thermopapier eingelegt (s. u.) und die System-Stromversorgung eingeschaltet haben, geben Sie ein: ADR – 2011, RUN; den Rest verfolgen Sie dann gebannt; denn Ihr Drucker meldet artig den vollbrachten Erfolg, indem er sagt: „Gratulation, er geht" (nur wenige Bausätze sind derart wohlerzogen). Wenn Sie das schick finden, teilen Sie ganz und gar unsere Meinung; wir sagten doch: Er mausert sich, der MOPPEL!
Ausdrucken können Sie nun auf zwei Arten; entweder wollen Sie sich einen Speicherabzug („HEX-Listing") ausgeben lassen, wo Adressen und (hexadezimaler) Inhalt gegenübergestellt werden (**Bild 56** oben). Oder Sie wollen Klartexte drucken (im Bild 56 unten), angefangen bei Meßwertreihen, über Visitenkarten bis hin zu Termin-Erinnerungen: „Heute ist der 23. Dezember, dringend an Weihnachtsgeschenke denken!" Ganz Schlaue erweitern dieses Programm auf die Wochentagsabfrage: „Es ist zwar höchste Zeit für Weihnachtsgeschenke, aber Sonntag, also zu spät!" Im Rahmen unserer Mikrocomputer-Seiten werden wir noch ausführlich auf eine solche Terminüberwachung

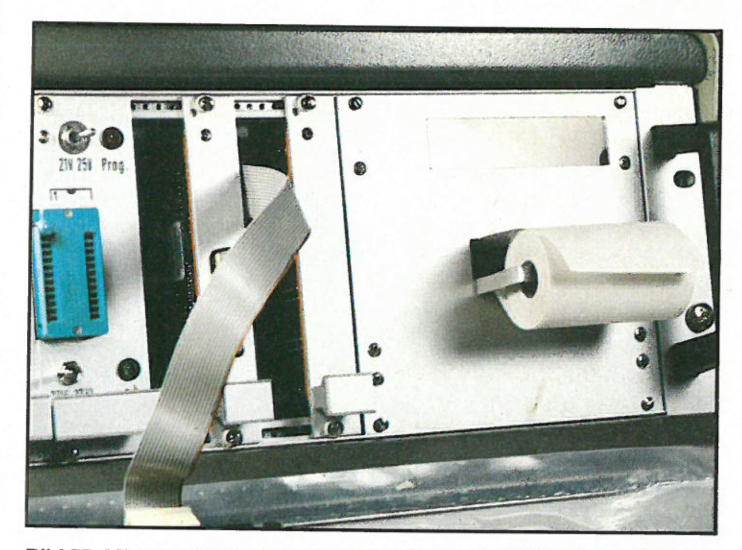


Bild 57: Mit dem mechanischen Zubehör der Profi-Version entsteht eine Einheit, die sich nahtlos in einen 19-Zoll-Einschub einfügt.

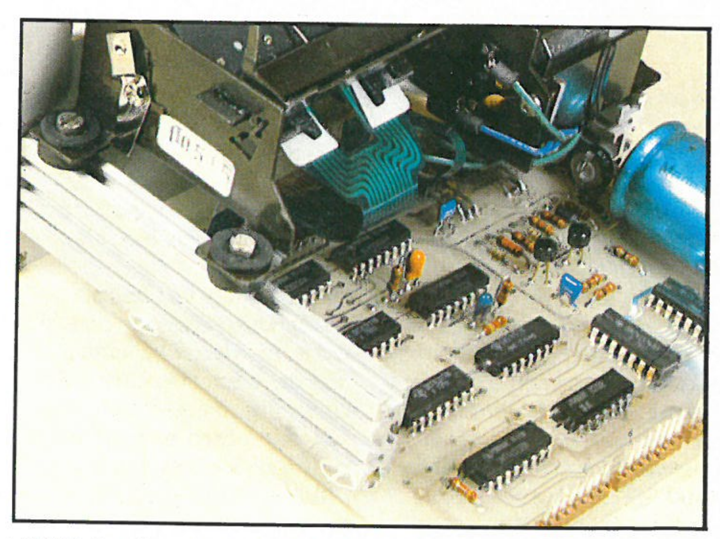


Bild 58: Druckwerk und Elektronik werden von oben und unten an die Profilschienen angeschraubt, in deren Hohlprofil die Schiebemuttern einzulegen sind.

eingehen, die ein (vielleicht *das* Parade-)Beispiel für einen sinnvollen Mikrocomputer-Einsatz ist.
Das HEX-Listing vollzieht sich wie folgt: Eingabe von FCT – D (Drucken), gefolgt von der Startadresse, NXT, der Endadresse, NXT und abgeschlossen von „EE" (für HEX-Ausdruck), RUN; das Programm rundet dann die eingegebenen Adressen auf „gerade" Werte und bringt diese, sauber sortiert, zusammen mit dem jeweiligen Inhalt zum Ausdruck. Für den Klartext-(ASCII-)Ausdruck geben Sie wiederum ein: FCT – D, Startadresse (plus NXT) sowie Zeilenanzahl (plus NXT), gefolgt von „AA" (für ASCII-Ausdruck), RUN; es werden dann (hexadezimal) so viele Zeilen mit jeweils 20 Zeichen ausgedruckt, wie Sie es in der zweiten Eingabe spezifiziert haben, wobei im angegebenen RAM-Bereich natürlich ASCII-Zeichen stehen müssen, weil ansonsten unsinnige Punktmuster aufs Papier gelangen (**Tabelle 5**)!

Für diese Montage müssen Druckwerk (oben) und Platine (unten) an die Profilschienen angeschraubt werden, die ihrerseits mit dem Stirngewinde an der Frontplatte verschraubt werden (**Bild 58**).
Da Sie bei dieser Anordnung nicht mehr ohne weiteres das Papier einfädeln können, helfen Sie sich mit folgendem Trick weiter: Sie füllen einen beliebigen Speicherbereich mit Nullen auf (FCT – A); dann lassen Sie diesen Bereich als ASCII-Zeichen ausdrucken (das ist nur lauter Luft, sprich Leerzeichen). Bei diesem Bemühen zieht der Drucker aber das ihm vorn am Schlund angebotene Papier ein, und drin ist es!

Achtung! Den Drucker dürfen Sie niemals ohne eingelegtes Papier betreiben; der kleine Hebel auf der Oberseite muß sich beim Papierzuführen im kraftlosen Zustand befinden, bei dem die Thermoköpfe auf das Papier drücken. Das Thermopapier ist ein Standard-Artikel aus dem Büro-Fachhan-

Tabelle 5. ASCII-Zeichenvorrat (American Standard Code For Information Interchange)

	X0	X1	X2	X3	X4	X5	X6	X7	X8	X9	XA	XB	XC	XD	XE	XF
2X		!	"	#	$	%	&	'	(	)	★	+	,	–	.	/
3X	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4X	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5X	P	Q	R	S	T	U	V	W	X	Y	Z	Ä	Ö	Ü	^	_
6X	\	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7X	p	q	r	s	t	u	v	w	x	y	z	ä	ö	ü	ß	■

Beispiel: „★" ist in Zeile 2 und Spalte A; der ASCII-Code lautet demnach „2A"

Der Drucker im vornehmen Gewand

In der Profi-Version des Thermodruckers sind mechanische Komponenten enthalten, mit denen sich die gesamte Baugruppe im 19-Zoll-Rahmen unterbringen läßt (**Bild 57**). Damit entsteht schon ein kleines Datenerfassungssystem, das auch für Protokolldrucke und Klartextbelege einzusetzen ist.

del; die Erstausstattung liegt dem Bausatz selbstverständlich bei.
Die Krönung der bisherigen Arbeiten bildet es, wenn Sie die Uhrzeit der (zuvor richtig gestellten) Echtzeit-Uhr auslesen und anschließend als Klartext ausdrucken lassen. Dazu benötigen Sie noch folgende Randinformationen: Das obenerwähnte Zeit-Unterprogramm TI (Startadresse 0061) legt den ASCII-Code für die Uhrzeit und Datum im RAM ab, von wo ihn sich das ASCII-Druckprogramm PRTASC (Startadresse 2003) holen muß, um ihn auszudrucken; der Zeilenzähler, der angibt, wie viele ASCII-Zeilen zu drukken sind, befindet sich in RAM-Zelle 2FD3; in diesem Fall ist er mit 01 (= eine Zeile) zu laden.
Mit folgender Sequenz können Sie sich Uhrzeit und Datum auf Knopfdruck ausdrucken lassen: CALL TI (Zeit holen und im ASCII-Code im RAM ablegen); MVI A, 01; STA 2FD3 (Zeilenzähler = 1); CALL PRTASC (Information ausdrucken); HLT. Im HEX-Code sieht das Ganze wie folgt aus: CD 61 00, 3E 01, 32 D3 2F, CD 03 20, 76. Machen Sie die Probe aufs Exempel und drükken Sie nach dem Laden RES und RUN; der ordnungsgemäße Ausdruck wird der Lohn Ihrer Mühe sein!
Manche mögen es gern sehen, wenn ihr Drucker sich die Zeit mit dem Ausdrucken seines Zeichenvorrats vertreibt (**Bild 59**); das sieht so wohlgeordnet aus, daß man es sich übers Bett hängen kann, und wenn Sie Sehnsucht danach haben, starten Sie das zugehörige Programm ab 2009; beenden können Sie diese Fließbandarbeit wie eben beschrieben.
Wenn Sie nun den Zeichenvorrat Ihres Druckers verändern oder erweitern möchten, stellen Sie das folgendermaßen an: Sie kopieren das gesamte orange Drucker-EPROM in den Adreßbereich 8000...87FF des RAMs (auf der großen Speicherkarte); dann denken Sie sich das Punktraster für Ihr neues Zeichen aus und laden die sieben zugehörigen Bytes in die RAM-Zellen 2810...2816. Am Beispiel des Ohm-Zeichens zeigt Ihnen **Bild 60** die Vorgehensweise, bei der in jeder Spalte ein zu druckender Punkt einem HIGH-Bit entspricht. Mit dem (ebenfalls im orangen Drucker-EPROM enthaltenen) Codierungsprogramm CODE (Adresse 2006) verteilen Sie den auf diese Weise geladenen Code in die Punktrastertabelle des Druck-Programms; *dazu müssen Sie vor Aufruf von CODE das Register A mit demjenigen ASCII-Wert laden, unter dem das neu kreierte Zeichen wiederzufinden sein soll.* Angenommen, Sie wollen dem Zeichenvorrat tatsächlich das Ohm-Zeichen einverleiben, und zwar unter dem ASCII-Code 80HEX (freier Bereich: 80...9F); dann laden Sie das Punktmuster gemäß Bild 60 in die RAM-Zellen 2810...2816, laden Reg A mit 80 (ASCII-Code) und rufen dann CODE auf: MVI A, 80; CALL CODE; HLT (3E 80, CD 06 20, 76). Danach ist das

Bild 59: Mit dem ebenfalls im Drucker-EPROM enthaltenen Test-Programm können Sie sich meterweise den Zeichenvorrat ausgeben lassen.

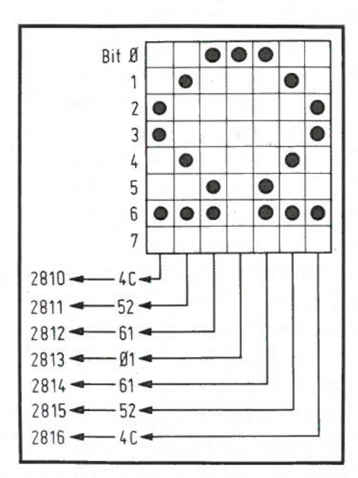


Bild 60: Sie können den Zeichenvorrat Ihres Druckers sogar erweitern oder modifizieren; hier das Beispiel für das Ohm-Zeichen.

Punktraster gezielt so in der Tabelle verteilt, daß der Drucker später, wenn er das ASCII-Zeichen „80“ angeboten bekommt, daraus ein „Ω“ formt. Nach dem Eingeben aller neuen Zeichen programmieren Sie den RAM-Bereich 8000...87FF zurück in ein (neues) 2716-EPROM und setzen dies in Platz #2 der CPU-Karte ein; Ihr Drucker versteht nun den erweiterten bzw. modifizierten Zeichensatz, wovon Sie sich mühelos überzeugen können.

MOPPELs „weiche“ ASCII-Tastatur

Die ASCII-Tastatur mit deutscher Normbelegung und der abgesetzte Zehnerblock mit Cursor-Steuerung bilden die Eingabeseite des bildschirmorientierten Mikrocomputer-Systems.
Bisher haben wir uns ausschließlich auf der untersten Maschinen-Ebene bewegt, wenn es um die Mikrocomputer-Programmierung ging. Zur Eingabe diente die HEX-Tastatur, auf der für die Ausgabe auch gleich noch eine Anzeige vorhanden war; wie Sie gesehen haben, können Sie mit dieser Ein-/Ausgabe-Einheit die bisher vorgestellten Peripherie-Baugruppen mühelos ansprechen (EPROM-Programmierzusatz, Echtzeit-Uhr und Thermodrucker). Zur Programmierung in einer höheren Programmiersprache (wie z. B. BASIC) reicht diese HEX-Tastatur und -Anzeige natürlich nicht mehr aus, da muß eine „richtige“ Schreibmaschinen-Tastatur her. Die gute alte HEX-Eingabe landet deshalb aber noch lange nicht auf dem Schrott: Zum Betrieb der später folgenden Funkuhr ist sie nämlich unerläßlich (dasselbe gilt für das Mini-Netzteil).
Mit der hier vorgestellten ASCII-Tastatur (**Bild 61**; das heißt „Aski“, weil es von *American Standard-Code for Information Interchange* kommt, und nur ganz Unbelehrbare bleiben bei der sinnlosen Aussprache „A-Es-Zeh-Zwo“) und dem folgenden Video-Interface vollziehen wir den Schritt zum bildschirmorientierten Computer-System, mit dem Sie (zumindest vom Gefühl her) ein waschechter Profi werden! Tun Sie uns gemeinsam einen Gefallen und haken Sie diese Tastatur nicht wie selbstverständlich ab, weil sich dahinter nichts Weltbewegendes verbirgt; denn die Tastatur auf der einen und der Bildschirm auf der anderen Seite sind schließlich genau *die* Schnittstellen, über die Sie sich mit Ihrem Maschinchen unterhalten können. Und weil jeder Kontakt über diese Schnittstellen abläuft, sollte bei der Auswahl und Gestaltung gerade dieser Komponenten höchste Sorgfalt verwendet werden. Versuchen Sie einmal, auf einer Knackfrosch-Tastatur längere Programme oder Texte einzugeben; oder biegen Sie Ihre Finger mal auf einer Folientastatur krumm, nie wissend, ob sie die Eingabe nun gefressen hat oder nicht. Die Preiswürdigkeit von Billig-Tastaturen wiegt deren mangelnde Bedienfreundlichkeit zu keinem Zeitpunkt auf.

Hier kontrolliert kein Controller

Für MOPPEL haben wir deshalb eine Tastatur mit hochwertigen und bedienfreundlichen Tasten ausgewählt, die in jeder Hinsicht dem modernsten Stand entsprechen; gespart haben wir da, wo es zulässig ist und nicht auf Kosten der Handhabung geht: Die Tastatur besitzt keinen teuren und stromfressenden Steuerbaustein (Ihnen geläufig unter der neudeutschen Bezeichnung „Keyboard Encoder“),

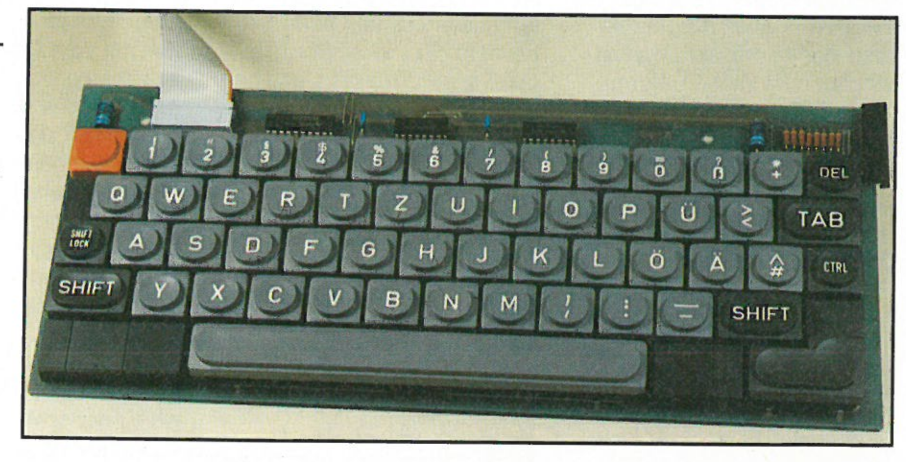


Bild 61: Bedienfreundlichkeit und hoher Qualitätsstandard zeichnen die MOPPEL-ASCII-Tastatur aus.

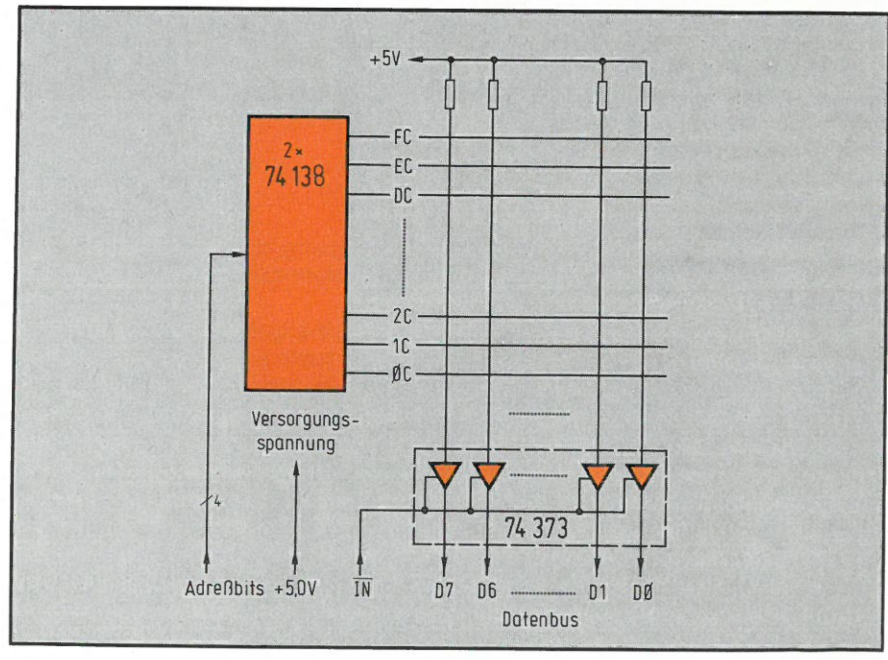


Bild 62: Die Tastatur ist in Form einer Matrix organisiert, an deren Kreuzungspunkten jeweils eine Taste liegt, die die jeweilige Zeilen- und Spaltenleitung kurzschließen kann.

sondern sie wird ziemlich direkt an den System-Datenbus angeschlossen. Diese Lösung ist nicht nur höchst elegant, sondern auch noch wahnsinnig preiswert. Und weil die ganze Tastatur-Abfrage per Programm (und ohne Steuerbaustein) geschieht, spricht man von einer Soft-Tastatur; das „Weich" in diesem Zusammenhang bezieht sich also nicht nur auf die federleichte Betätigungskraft, sondern auch noch auf die Software im Hintergrund.

Die Tastatur ist in Form einer Matrix mit 16 (waagerechten) Zeilen und 8 (senkrechten) Spalten organisiert, an deren Kreuzungspunkten jeweils eine Taste liegt; jede Spalte wird über einen Pull-up-Widerstand mit +5 V verbunden und über ein Tri-State-Gatter an ein Bit des Datenbus geführt (**Bild 62**); die acht Tri-State-Gatter (aus Platzgründen wird hierfür der 8-Bit-Speicher 74373 mit Tri-State-Ausgängen eingesetzt) werden immer dann aktiviert (d. h. auf den Datenbus geschaltet), wenn die CPU einen Ein-/Ausgabe-Befehl mit „XC" in der Portadresse ausführt. Dieses Aktivierungssignal $\overline{IN}$ entsteht in der Selektierungslogik auf der CPU-Platine. Die obere Hälfte der Portadresse (genauer gesagt: die vier Adreßbits A4...A7) gelangt an einen Demultiplexer, der aus zwei 74138 zusammengesetzt ist; von den 16 Ausgängen ist immer nur einer aktiv, d. h. auf LOW, und zwar derjenige, den die vier Adreßbits ansprechen. Die 8-Bit-Portadresse wird im Fall der Tastatur folglich in zwei Häppchen verarbeitet:

Die untere Hälfte „C" bewegt die Selektierungslogik dazu, den Aktivierungsimpuls $\overline{IN}$ zu erzeugen und damit den Pegel der acht Spaltenleitungen in die CPU einzulesen; die oberen vier Bits der Portadresse geben an, welche der Zeilenleitungen dabei auf LOW gehen soll. Zur Abfrage aller Tasten sind folglich 16 einzelne IN-Befehle auszuführen, um sämtliche 16 Zeilendrähte abzuklappern.

Ja, wo drücken Sie denn?

Wie kriegt man nun heraus, ob eine und, wenn ja, welche Taste gedrückt ist? Das ist nun nicht mehr sehr schwierig, wenn Sie sich folgendes überlegen: Ist keine Taste gedrückt, liegen alle Spaltenleitungen auf HIGH-Potential, so daß die CPU in diesem Fall „FF" einliest. Drückt man aber eine Taste, geht die zugehörige Spaltenleitung in dem Augenblick auf LOW, wo die an der betreffenden Taste liegende Zeilenleitung LOW ist. Folglich hat sich der Mops unaufhörlich damit zu plagen, nacheinander die 16 Einlesebefehle IN OC...IN FC auszuführen und jedesmal danach zu fragen, ob eines der Datenbits auf LOW liegt; ist dies der Fall, erkennt das Programm aus der Bitposition und der gerade erzeugten Portadresse, um welche Taste es sich handelt. Mit diesem Wissen greift das Programm in eine Tabelle, in der die ASCII-Codes zusammengefaßt sind, holt das passende Bitmuster heraus und tut anschließend so, als stamme diese Information von einem intelligenten Steuerbaustein. Im Gegensatz zu diesem können Sie hier aber jede beliebige Codierung erzeugen, da die ja im EPROM (des roten Monitors) abgelegt ist. Und so ist es auch ein Kinderspiel, daß auf der MOPPEL-Tastatur „Y" und „Z" da sind, wo sie (nach unserem Empfinden) hingehören, nämlich links unten bzw. in der Mitte oben und nicht umgekehrt, wie bei den Geräten amerikanischen Ursprungs. Daß sich mit diesem Konzept mühelos auch solche Feinheiten wie "Ä", „Ö", „Ü" und „ß" unterbringen lassen, sei nur am Rande erwähnt, denn für eine deutsche Tastatur sollte der deutsche Zeichensatz eine Selbstverständlichkeit sein (**Tabelle 6**)! Die farblich hinterlegten Codes werden beim gleichzeitigen Druck auf SHIFT (grün) bzw. CONTROL (blau) erzeugt; den rotmarkierten Tasten sind unabhängig von SHIFT oder CONTROL immer dieselben Codes zugeordnet. Daß diese Tabelle so scheinbar lieblos und willkürlich mit Zeichen gefüllt ist, liegt an zwei Dingen: Erstens sind bestimmte Symbole auf einer Taste nach Sinnfälligkeit der Bedienung und nicht nach Logik der Code-Generierung kombiniert; und zweitens sind die neu aufgenommen Sonderzeichen so verteilt worden, daß ihre Abfrage (und Umcodierung für das Video-Interface) mit möglichst geringem Software-Aufwand erfolgen kann.

Mit den Tasten des Zehnerblocks auf der Tastatur-Erweiterung hat es übrigens eine besondere Bewandtnis: Unabhängig von der normalen oder hochgestellten (SHIFT-)Betriebsart erzeugen diese Zifferntasten stets die entsprechende Ziffer; mit der grünen FUNCTION-Taste werden sie zu frei programmierbaren Funktionstasten (s. u.), die auf eine weitere Ebene frei programmierbarer Tasten umschaltbar sind, indem Sie an Stelle der FUNCTION- die CONTROL-Taste drücken. In diesem Fall müssen Sie den zugehörigen Sprungverteiler im RAM anlegen (ab Adresse 37D0, untere Adreßhälfte zuerst).

Tanz in drei Etagen

Beim Blick auf **Bild 63** erkennen Sie weitere Feinheiten der Organisation: Die Umschalt-

Tabelle 6: Codes der MOPPEL-ASCII-Tastatur

	XF	XE	XD	XC	XB	XA	X9	X8	X7	X6	X5	X4	X3	X2	X1	XØ
FX		•					9	8	7	6	5	4	3	2	1	Ø
EX																
DX																
CX																
BX	}	{	]	[	“	„	|	≈	√	‰	∢	±	^{3}	^{2}	^{1}/	^{0}
AX	**O**	**L**	**E**	Ω	τ	Σ	π	μ	λ	ϰ	ε	δ	γ	β	α	@
9X															‘	’
8X	↖				↓		↑							→	DEL	←
7X		ß	ü	ö	ä	z	y	x	w	v	u	t	s	r	q	p
6X	o	n	m	l	k	j	i	h	g	f	e	d	c	b	a	
5X	_	^	Ü	Ö	Ä	Z	Y	X	W	V	U	T	S	R	Q	P
4X	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A	§
3X	?	>	=	<	;	:	9	8	7	6	5	4	3	2	1	Ø
2X	/	.	-	,	+	*	)	(		&	%	$	#	"	!	␣
1X					ESC CTLC						NAK CTLX		DC3 CTLW		DC1 CTLQ	
0X	SI CTLX	SO CTLY	RET				TAB		BEL CTLR							

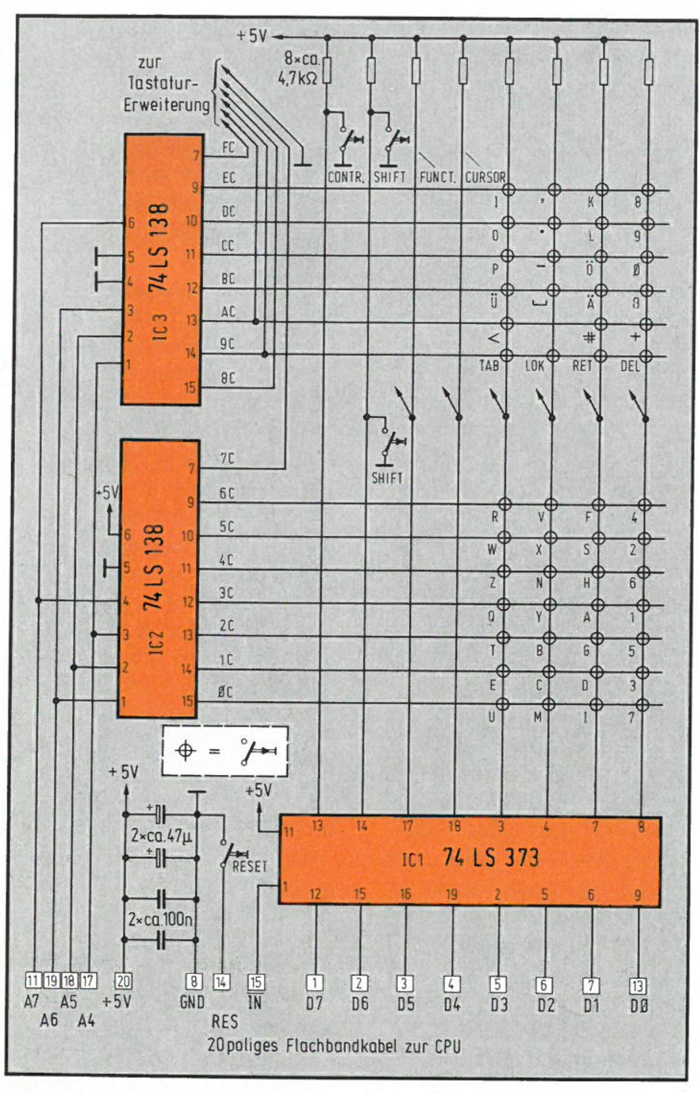


Bild 63: Dank des gewählten Schaltungskonzepts kommt die Tastatur ohne aufwendigen Steuerbaustein aus.

tasten CONTROL, SHIFT und FUNCTION bekommen exklusiv eine eigene Spaltenleitung zugeordnet (Datenbits 7, 6 und 5); mit diesen Tasten erfolgt die Umschaltung in eine andere Ebene, d. h. je nach Aktivierung einer der Umschalttasten gibt eine Zeichentaste unterschiedliche Codes her. An die Datenleitung 4 sind die fünf Steuertasten zur Cursor-Bewegung angeschlossen, die (wie auch die FUNCTION-Taste FCT) auf der Tastatur-Erweiterung angeordnet sind. Und die restlichen vier Datenleitungen 0...3 bilden die Spaltenleitungen der eigentlichen Tasten-Matrix (**Bild 64** und **65**). Die SHIFT-Taste macht genau dasselbe wie die Hochstell-Taste an Ihrer Schreibmaschine: Wird sie zusammen mit einer anderen Taste betätigt, wird ein Großbuchstabe (bzw. das obere von zwei Symbolen auf einer Taste) erzeugt; um diesen Zustand zu verrammeln (d. h. fest einzuklinken), können Sie SHIFT LOCK drücken. Zum Entrammeln (d. h. wieder Ausklinken) betätigen Sie diese Taste erneut (Flipflop-Funktion). Die CONTROL-Taste schaltet die Tastatur um auf eine dritte Ebene, in der Sonderzeichen und spezielle Steuerzeichen ausgegeben werden (s. u.). FUNCTION schließlich dient dazu, den Zehnerblock der Tastatur-Erweiterung auf frei programmierbare Funktionstasten umzuschalten. Die Sprungtabelle hierzu muß im RAM ab Adresse 37E8ff. angelegt werden. Dort stehen die Zieladressen, zu denen verzweigt werden soll (unteres Byte immer zuerst), und zwar in folgender Reihenfolge: FCT 9, 3, RET, 6, 7, 1, 0, 4, 8, 2, Punkt und 5. Diese Funktionsumschaltung können Sie mit Sprungzielen

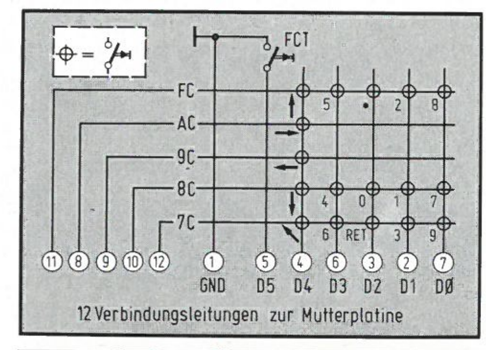


Bild 64: Für die professionelle Ausstattung vorgesehen sind die Tasten zur Cursor-Steuerung, ein abgesetzter Zehnerblock und die Funktionsumschaltung.

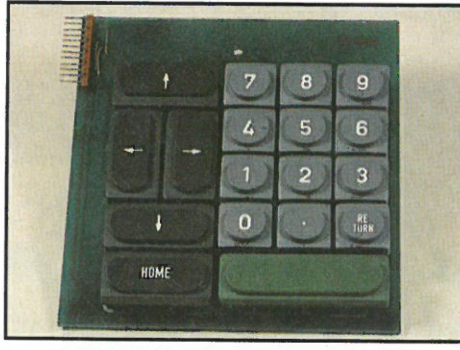


Bild 65: Die Tastatur-Erweiterung vervollständigt die auf der Mutterplatine angeordnete Tastenmatrix.

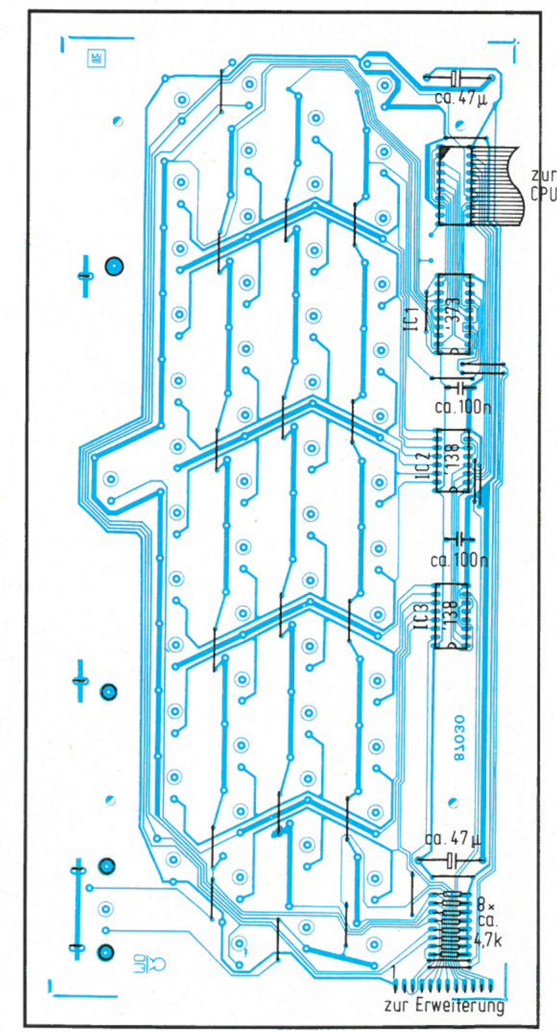


Bild 66: Vor dem Einlöten der Tastenkörper sind die hier eingezeichneten Bauteile zu bestücken; ganz unten sind die Führungshülsen und Haltebügel für die breiten Tasten einzusetzen.

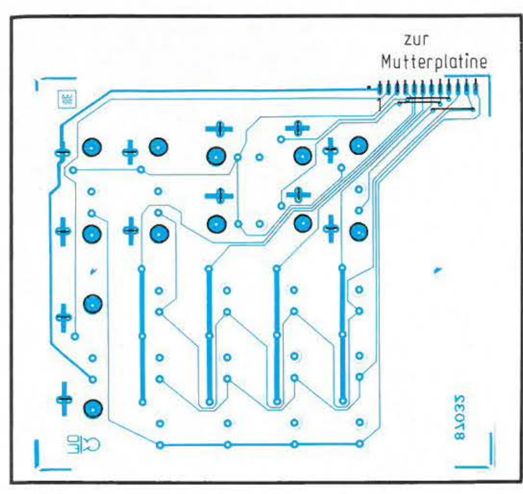


Bild 67: Außer den drei Drahtbrücken sind hier die mechanischen Teile für fünf Tasten mit Überbreite zu montieren.

Ihrer Wahl versehen, was natürlich nicht identisch sein muß mit den Funktionstasten der HEX-Tastatur.

Weil der Nachbau so einer simplen Tastatur augenscheinlich ganz von allein geht, erübrigt sich jedes weitere Wort; das zumindest denkt der Pfuscher, der bereit ist, sein Werk später ohne jeden Groll wegzuschmeißen. Wenn Sie das nicht vorhaben, lesen Sie die folgenden Zeilen vor dem Zusammenbau getrost durch (was sich beim Kauf der fertig montierten Tastatur natürlich erübrigt). Beginnen Sie die Bestückung mit den 27 Brücken auf der Mutterplatine (und gegebenenfalls den drei Brücken auf der Erweiterungsplatine), gefolgt vom Einlöten der Widerstände, Kondensatoren und ICs (**Bild 66** und **67**). Danach folgt von der Bauteilseite her das Einsetzen der schwarzen Buchsen, die bei den breiten Tasten als Lager für die weißen Stößel dienen (vier Stück auf der Mutter- und 12 Stück auf der Tochterplatine **Bild 68**). Und schließlich löten Sie mit Hilfe von jeweils zwei Haltebügeln die Drahtbügel ein, die in die Stößel eingreifen (**Bild 69**). Bitte wenden Sie keinerlei Gewalt an, wenn Sie den Drahtbügel rechts und links in die Stößel einfädeln, es geht mit ruhiger Hand und etwas Geschick vollkommen kraftlos (**Bild 70**).

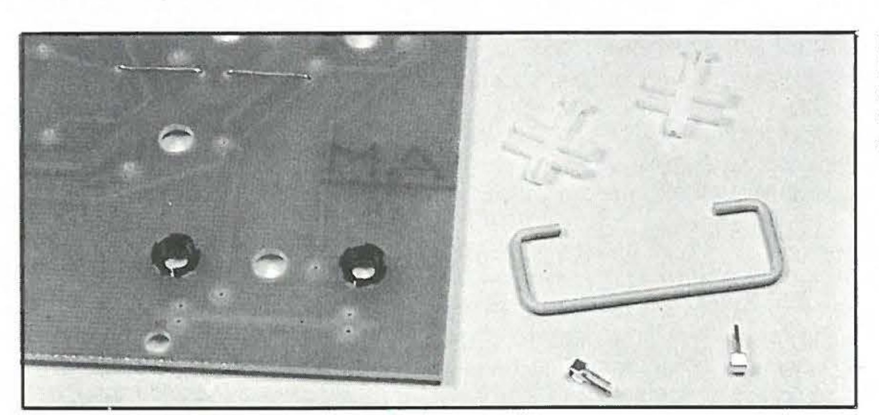


Bild 68: Von der Bauteilseite her erfolgt der Einsatz der Führungshülsen für die Stößel.

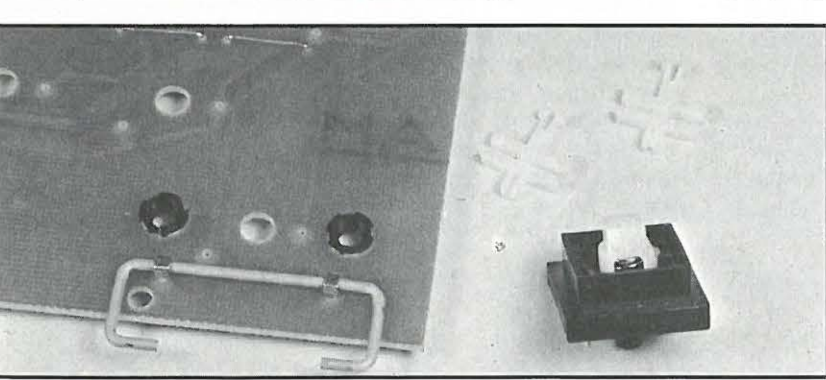


Bild 69: Jeweils ein Paar von Haltebügeln führt den Drahtbügel, der bei den breiten Tasten die beiden Stößel miteinander verbindet.

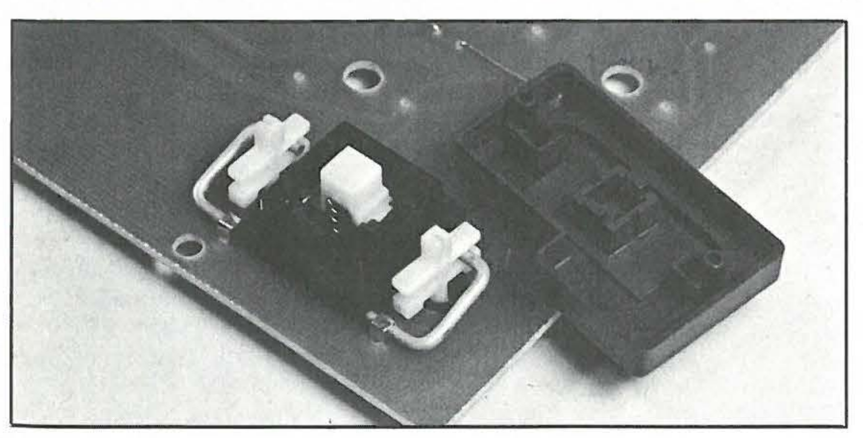


Bild 70: Die Vormontage wird abgeschlossen durch das beidseitige Einführen des Drahtbügels in die Stößel.

Auf jede Taste paßt ein Deckel

Erst nach Abschluß und sorgfältiger Kontrolle dieser Arbeiten kommt das Einlöten der Tastenkörper an die Reihe. Das geht Stück für Stück vor sich, wobei Sie jedes einzelne Exemplar beim Löten fest an die Platine andrücken; nur so ist gewährleistet, daß hinterher keine schiefen Köpfe aus einer Kraut-und-Rüben-Tastatur herausragen! Wenn Sie damit fertig sind, trinken Sie am besten eine Tasse Kaffee oder tun etwas Vergleichbares, denn das Aufsetzen der Kappen sollte erst recht mit Verstand (und der nötigen Ruhe) erfolgen, weil sich so eine Kappe gegen das Abziehen aufs Heftigste sträubt (**Bild 71** und **72**); das Resultat ist dann in den meisten Fällen ein defekter Unterbau, was man sich durchaus ersparen kann; die Kappen müssen übrigens fest auf ihre Führungsnase aufgedrückt werden, bis sie merklich einrasten.
Für vier Tasten sind wir Ihnen noch die Erklärung schuldig, hier ist sie: DELETE löscht das zuletzt eingegebene Zeichen wieder aus, und zwar so, als sei es nie dagewesen; im Gegensatz zum Ausdruck auf Papier funktioniert das auf dem Bildschirm prächtig! Beim Druck auf die TAB-(ulator-)Taste rückt der Cursor („Körßer"; er markiert auf dem Bildschirm diejenige Stelle, an der die nächste Eingabe landet) kommentarlos um acht Stellen nach rechts. Um die RETURN-Taste, die sich bescheiden ganz unten rechts befindet (als Winkeltaste auf der Mutterplatine und entsprechend beschriftet auf der Erweiterung),

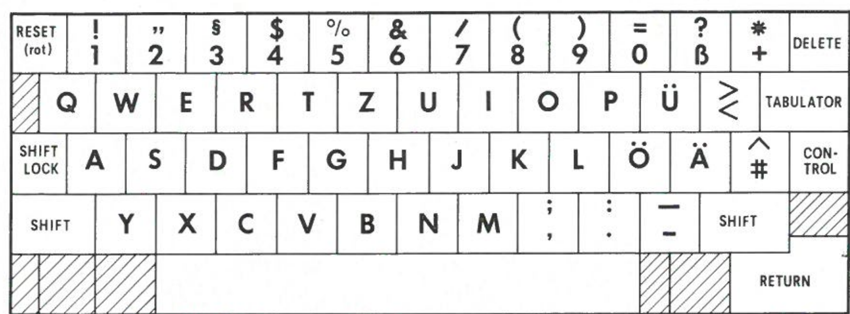


Bild 71: Selbstverständlich sind die Tasten entsprechend der deutschen Normbelegung angeordnet (die schraffierten Flächen sind Füllstücke).

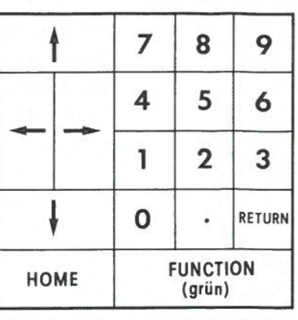


Bild 72: Mit der FUNCTION-Taste läßt sich der Zehnerblock derart umschalten, daß zehn frei programmierbare Funktionstasten entstehen.

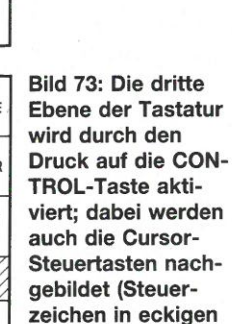


Bild 73: Die dritte Ebene der Tastatur wird durch den Druck auf die CONTROL-Taste aktiviert; dabei werden auch die Cursor-Steuertasten nachgebildet (Steuerzeichen in eckigen Klammern sind nicht darstellbar).

hat eine Doppelfunktion; ihr eigentlicher Zweck ist es, auf den Anfang einer neuen Zeile umzuschalten (wie Sie es von der Schreibmaschine her kennen). Darüber hinaus ist diese Taste so etwas wie ein hauptberuflicher Auslöser, denn mit ihr werden später bestimmte Vorgänge ausgelöst, angefangen beim Start von Anwender-Programmen bis hin zum Abschließen bestimmter Betriebsarten. Die vierte übriggebliebene Taste HOME hat etwas Anheimelndes an sich: Sie führt nämlich den Cursor in die linke obere Ecke des Bildschirms, wo er zu Hause ist (merke: Des Cursors Heim ist keine Festung, sondern eine Ecke). Da dies auch der Anfang des Bildwiederholspeichers ist, mag das mit dem Zuhause in Ordnung gehen, auch wenn Ihnen dafür vielleicht eine andere Ecke lieber wäre!

Um die im Zeichensatz des Video-Interfaces enthaltenen Sonderzeichen aktivieren zu können, schalten Sie das Tastenfeld mit CONTROL um: Sie können dann die in **Bild 73** dargestellten Symbole erzeugen und außerdem ein paar Steuerzeichen aufrufen (die in eckigen Klammern stehenden), auf deren Bedeutung wir später noch eingehen. Auch ohne die Tastatur-Erweiterung können Sie Ihren Cursor über den Schirm flitzen lassen, wenn Sie die fünf zentral liegenden Tasten H, N, J und Z mittels CONTROL umschalten; sie ersetzen dann die Pfeiltasten zur Cursor-Steuerung.

Wenn Sie eine REPEAT-Taste vermissen (die macht Dauerfeuer, d. h. sie wiederholt permanent das zuletzt eingegebene Zeichen), dann seien Sie getröstet: Die MOPPEL-Tastatur hat auch einen Soft-REPEAT („Riepiet"); wenn Sie eine Taste etwas länger gedrückt halten, wird die Wiederhol-Funktion automatisch eingeschaltet, und die bleibt dann so lange aktiv, bis Sie wieder loslassen.

Wie Sie sehen, verbirgt sich hinter so einer Tastatur doch eine ganze Menge mehr, als man beim ersten Hinsehen vermutet, ja, es ist letztlich eine ganze Philosophie, die zu dem Kompromiß aus Anzahl von Tasten, Umschalteebenen und Bedienfreundlichkeit führt und bei dem hier auf eine Super-SHIFT-Taste bewußt verzichtet wurde. Das gewählte Konzept hat sich in monatelangen Tests bestens bewährt und dürfte Garant dafür sein, daß Sie eine ermüdungsfreie und optimal gestaltete Eingabeseite vorfinden, wenn Sie darangehen, Ihrem Bildschirm-MOPPEL etwas mitzuteilen.

Video-Interface: Schaltung mit Fernblick

Das MOPPEL-Video-Interface kann gleichzeitig einen Standard-Fernseher, einen TV-Monitor und ein Mini-Sichtgerät ansteuern. –

Es gibt nur einen Weg, über den Sie sich mit Ihrem Computer verständigen können, und das sind die Ein-/Ausgabe-Einheiten; d. h. der spärliche „Dialog" zwischen Benutzer und Maschine beschränkt sich auf eine Ebene, auf der sich beide treffen können. Eingabeseitig ist dies in den meisten Fällen eine Tastatur, über die Sie im Klartext eintippen, welche Wünsche und Anweisungen Sie haben; und ausgabeseitig kennen Sie zur Genüge die vielsagenden Bildschirme, die allerorten die sich häufenden Computer zieren. So ein Bildschirm ist natürlich schon eine ganz andere Sache als eine simple Siebensegmentanzeige, weil da erstens viel mehr drauf paßt und zweitens die Zeichen so aussehen, wie es sich ein vernünftiger Mensch vorstellt: Da ist eine Drei natürlich wohlgerundet und nicht hölzern aus fünf Leuchtbalken zusammengesetzt, und selbstverständlich lassen sich auf dem Schirm sämtliche Zeichen darstellen, was mit sieben (oder auch 16) Segmenten eben einfach nicht

mehr möglich ist. Folglich hat ein voll ausgebauter Computer einen Bildschirm zu haben, über den er Sie wissen läßt, was er von Ihren Anweisungen hält (**Bild 74**). Und da kein Computer der Welt so einen „Fernseher" direkt ansprechen kann, benötigt er eine entsprechende Zwischenstufe, einen Übersetzer quasi, der die Computersprache in darstellbare Zeichen umsetzt; und welche Bezeichnung liegt hierfür näher als *Video-Interface*, womit wir gleich beim Thema sind (**Bild 75**). Wir halten an dieser Stelle bitte fest, daß ein Video-Interface eine Baugruppe ist, die die *von* einem Computer stammenden Informationen derart aufbereitet, daß sie auf einem Bildschirm darstellbar sind; außerdem sorgt dieses Interface dafür, daß die Daten permanent auf dem Schirm erscheinen, was sich bedeutend einfacher anhört, als es in der Praxis ist.

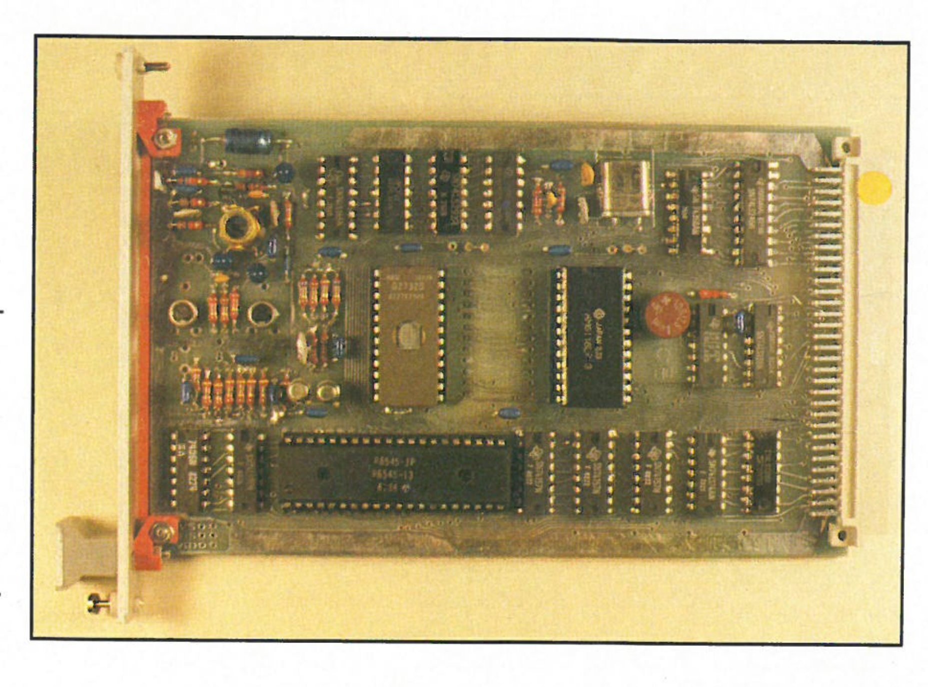


Bild 75: Ein hochintegrierter Steuerbaustein übernimmt die gesamte Koordination der Aktivitäten zur Bildaufbereitung und -darstellung (oben links ist der UHF-Modulator zum direkten Fernseher-Anschluß zu erkennen).

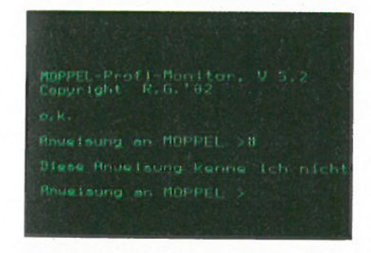


Bild 74: Der Bildschirm ist schon ein recht komfortables Verständigungsmittel zwischen Computer und Bediener.

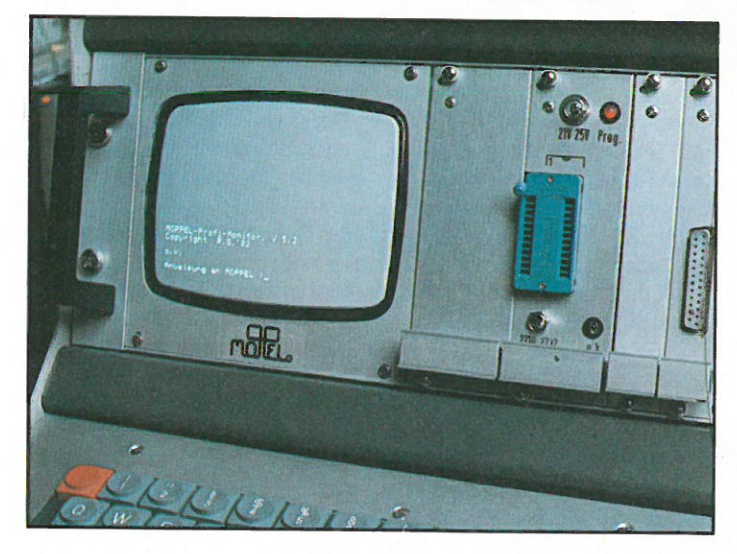
Neuzeitliche Sprachverwirrung

Bei den Begriffen „Bildschirm", „Monitor" und „Datensichtgerät" treten im Zusammenhang mit Computern häufig heillose Verwirrungen auf, die wir von vornherein ausräumen wollen: Fernseher ist das, was geeignet ist, die von Anstalten des öffentlichen Rechts ausgestrahlten, hochfrequenten Sendungen zu empfangen, zu demodulieren und auf dem Schirm sichtbar zu machen; Computer-Daten kann so ein Ding von Haus aus *nicht* darstellen (dazu ist neben dem eigentlichen Interface ein Modulator notwendig, der auf dem MOPPEL-Interface aber noch Platz hat). Unter „TV-Monitor" versteht man ein fernsehähnliches Gebilde, das aber keinen Empfangsteil besitzt und demzufolge sein Lebtag lang nie Sendungen wie „Dallas" oder Schlimmeres wiedergeben kann; ein TV-Monitor dient ausschließlich zur Datendarstellung, und er kann daher nur unmittelbar von einem Video-Interface angesteuert werden (den Vorsatz „TV" stellen wir konsequent zur Unterscheidung zum *Monitor*-Programm im EPROM voran). Im Rahmen unserer Namensgebung unterscheiden wir noch ein drittes Bildschirmgerät, nämlich unser Mini-Sichtgerät (**Bild 76**), das im Prinzip auch ein TV-Monitor ist, das im Gegensatz zum Normalfall aber mit Gleichspannung gespeist wird; das ist für den Betrieb eine wesentliche Randbedingung, weil bei Gleichstromversorgung die 50-Hz-Netzfrequenz zur Synchronisation fehlt, und die Ansteuerung daher anders aussehen muß. Das hier beschriebene Interface kann, wie gesagt, alle drei Geräte-Arten *gleichzeitig* versorgen, weil die drei dafür erforderlichen Ausgänge getrennt herausgeführt sind.

Ein Blitz-Überblick

Bevor wir ins Eingemachte gehen, speichern Sie bitte ein paar Fakten ab, die wir anschließend näher erläutern. Dieses Video-Interface ist für den bildschirmorientierten MOPPEL konzipiert; außerdem sind für den Bildschirm-Betrieb die ASCII-Tastatur sowie das rote *und* gelbe Monitor-EPROM erforderlich. Werfen Sie beim Lesen der folgenden Zeilen stets ein Auge auf **Bild 77**, das Ihnen einen groben Überblick über die Organisation unseres Video-Interfaces gibt. Auf dieser Baugruppe spielt der Bildwiederholspeicher die erste Geige; das ist ein RAM, das mit in den Adreßraum des Mikrocomputers eingebaut ist (vgl. Bild 11) und den 4-K-Bereich von 3000...3FFF belegt. Hiervon werden im Normalfall allerdings nur 2 K bestückt. In diesem Bildwiederholspeicher stehen (im ASCII-Code) diejenigen Zeichen, die auf dem Bildschirm erscheinen sollen; sie werden von der CPU dort hineingeschrieben und anschließend vom Video-Steuerbaustein herausgeholt, da-

Bild 76: Das Mini-Sichtgerät kann nicht mit dem zusammengesetzten BAS-Signal angesteuert werden, sondern es wird am Video-Interface an eigens hierfür vorgesehene Ausgänge angeschlossen.



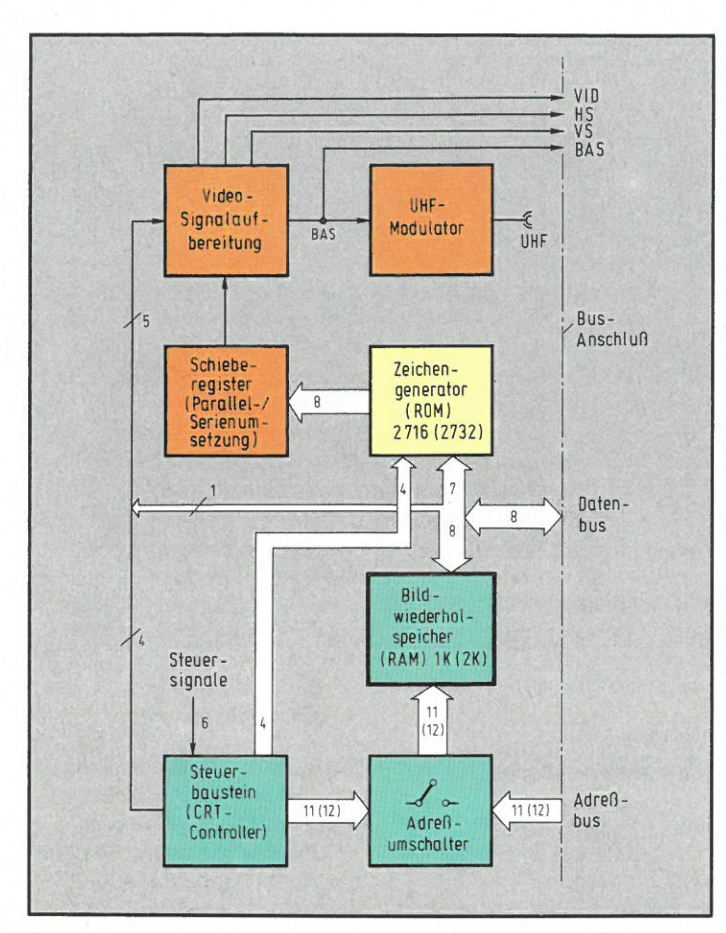


Bild 77: Schemazeichnung zur Funktionsbeschreibung des Video-Interfaces.

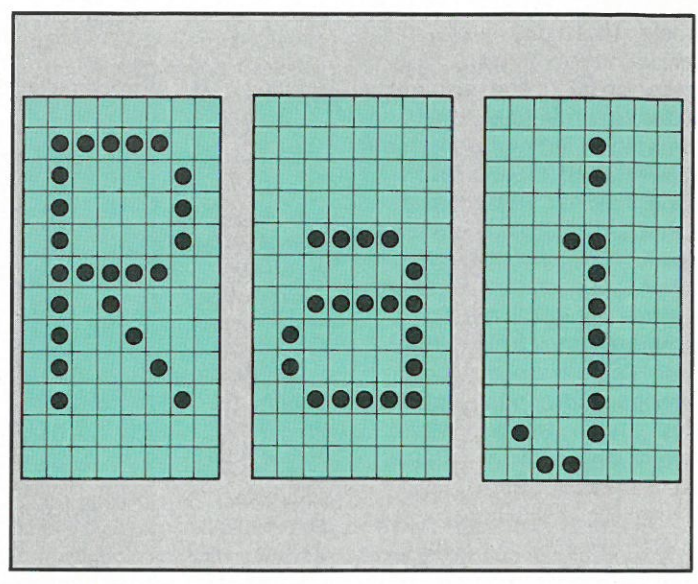
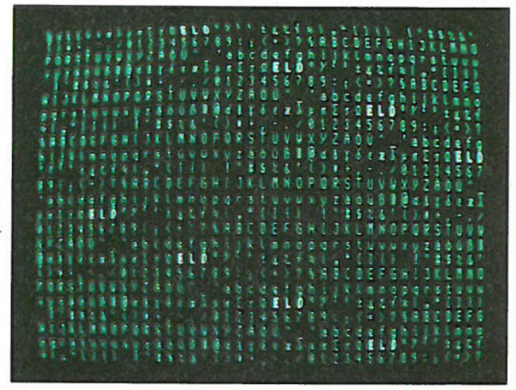


Bild 78: Durch die hohe Auflösung von 8 × 12 Punkten pro Zeichen wird eine sehr gute Lesbarkeit erreicht; dazu gehören selbstverständlich auch Unterlängen bei „g“, „j“, „p“, „q“ und „y“.



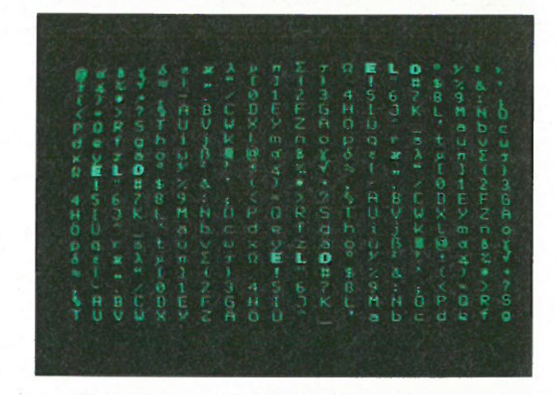


Bild 79: Die Hardware ist so gestaltet, daß zwei unterschiedliche Bildformate (24 × 80 oder 18 × 40 Zeichen) dargestellt werden können.

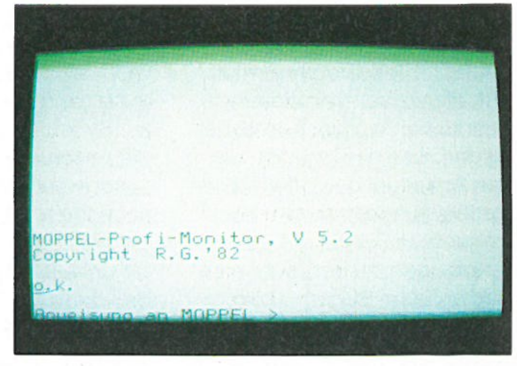


Bild 80: Durch Umverdrahten einer einzigen Leitung läßt sich die Bilddarstellung invertieren.

mit sie nacheinander der Bildaufbereitung zugeführt werden können. Auf dieses RAM haben folglich zwei Stellen Zugriff: Einerseits die CPU, die die Zeichen hineinschreibt (und sie bei Bedarf auch ohne weiteres wieder lesen kann) und andererseits der Steuerbaustein, der sich fortwährend damit plagt, die im RAM gesammelten Zeichen zur Anzeige zu bringen. Im Normalfall steht der Adreßumschalter so, daß Steuerbaustein und RAM gekoppelt sind; nur in dem Augenblick, in dem die CPU auf den Bildwiederholspeicher zugreift (Schreib- oder Lese-Operation im Adreßbereich 3000...3FFF), hat sie den Vorrang, d. h. der Adreßumschalter schaltet in diesem Fall automatisch auf den CPU-Adreßbus um.
Die Zeichendarstellung erfolgt in Form einer 8 × 12-Punkt-Matrix, d. h. jeder Buchstabe kann maximal 8 Punkte breit und 12 Punkte hoch sein. Um ein vernünftiges Schriftbild zu erhalten, sind die beiden unteren Punktreihen für Unterlängen reserviert (im Fachjargon „echte“ Unterlängen genannt), und die obere Reihe bleibt frei als Zwischenraum zur darüberliegenden Textzeile (**Bild 78**). Hardwarebedingt kann immer nur ein Block mit dem genannten Format von 8 × 12 Punkten als *Einheit* angesprochen werden. Innerhalb dieses Blocks ist jedes Muster möglich, einschließlich Grafik-Zeichen, nur kann man eben nicht einen einzelnen Bildpunkt adressieren (das wäre hochauflösende Grafik, die erheblich mehr Aufwand erfordern würde).
Die Schaltung kann zwei unterschiedliche Darstellungsformate erzeugen: Großformat mit 24 Zeilen zu je 80 Zeichen oder Kleinformat mit 18 Zeilen zu je 40 Zeichen (über Brükken wählbar; **Bild 79**). Wir haben (aus gutem Grund, wie wir meinen) zur Darstellung helle Schrift auf dunklem Hintergrund gewählt; es gibt Ergo-

nomen (das sind Leute, die sich dafür halten), die auf dunkle Schrift mit erhelltem Hintergrund bestehen, getreu dem Motto „Was man schwarz auf weiß besitzt..." Weil das an aggressiver Helligkeit aber kaum zu überbieten ist, bleiben wir bei Weiß (bzw. Grün) auf Schwarz, und wenn Sie eine Vorliebe für das Gegenteil haben, nehmen Sie von IC 6 nicht den Ausgang 7, sondern legen die Leiterbahn um auf Stift 9 (**Bild 80**)! Aufgabe des 40poligen Steuerbausteins ist es, ein Zeichen nach dem anderen aus dem Bildwiederholspeicher auszulesen (nachdem die CPU ihre Informationen dort hineingeschrieben hat) und zu veranlassen, daß jedes Zeichen pünktchenweise auf den Schirm kommt (**Bild 81**). Was das im einzelnen an Verwaltungsaufwand bedeutet, werden Sie gleich noch genauer sehen.

Bild 81: In der Vergrößerung sind auf dem Schirm die Pünktchen erkennbar, aus denen sich die einzelnen Zeichen zusammensetzen.

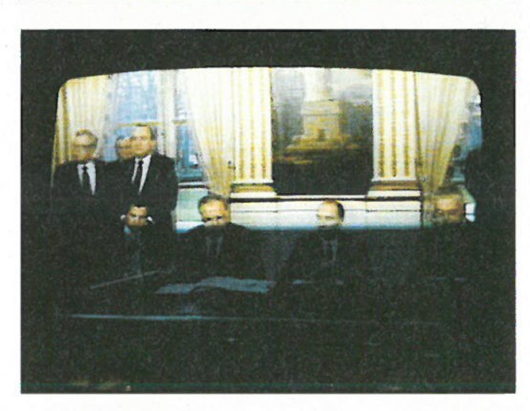


Bild 82: Eine zu kurze Belichtung macht deutlich, daß ein Fernsehbild immer nur teilweise auf dem Schirm vorhanden ist und daß es erst durch Nachleuchten und die Trägheit des Auges zu einem Ganzen wird.

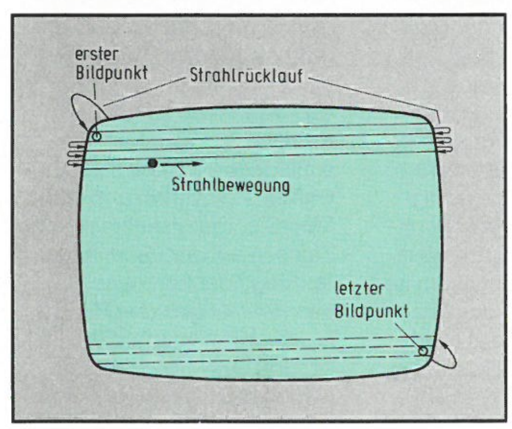


Bild 83: Auf seinem Weg über den Schirm überstreicht der Elektronenstrahl eine Bildzeile nach der anderen.

Ein Grundkurs in Fernsehen

Fernsehen kann eigentlich jeder, und natürlich können Sie sagen „her mit dem Video-Interface", es anschließen und fertig; dann aber entgehen Ihnen in Ihrem Leben zwei wesentliche Dinge: Erstens wissen Sie nie, welcher Aufwand für so ein simples, stehendes Bild erforderlich ist. Und zweitens bleibt es Ihnen dann auch für immer verborgen, wie sich der Steuerbaustein auf Ihrer Interface-Karte quälen muß, um eben dieses simple Bild aufzubereiten. Darum wagen Sie getrost mal einen tieferen Blick hinter die Mattscheibe, die ja sowieso schon längst zum Mittelpunkt unseres Lebens geworden ist!
Bei der üblichen Bildschirmdarstellung hat man einen unglaublichen Kompromiß gewählt, der recht brauchbar funktioniert; unglaublich ist das Ganze deshalb, weil es trotz einer ganzen Reihe fadenscheiniger Annäherungen erstaunlich gut klappt. Man stellt ein ganzes Bild nicht auf einmal dar, sondern zerlegt es in winzig kleine Pünktchen, genauso wie bei einem Bild in Ihrer Tageszeitung; man nennt diesen Vorgang *Rasterung*, und je feiner die Pünktchen gewählt werden, desto mehr treten sie in den Hintergrund und desto besser ist die Wiedergabequalität. Bei der Bild*wiedergabe* geht man den umgekehrten Weg und setzt das gesamte (Schrift-)Bild aus einzelnen Pünktchen zusammen; wenn das schnell genug passiert, und jedes Pünktchen ein bißchen nachleuchtet, dann denkt unser träges Auge, das ganze Bild sei auf einmal da. Daß dies nicht so ist, zeigt Ihnen ein Bildschirmfoto, das Sie nur kurz genug belichten müssen; bei 10 ms Belichtungszeit beispielsweise hat Ihr Fotoapparat gerade ein halbes Fernsehbild aufgeschnappt, während das Auge die schnell aufeinander folgenden Bilder zu einer fließenden Einheit „verschmiert" (**Bild 82**).
Das Zusammensetzen der Pünktchen geht nun folgendermaßen vor sich: Ein unermüdlich ausgesandter Elektronenstrahl wird so abgelenkt (aus seiner Null-Richtung, nicht von seiner Arbeit!), daß er zeilenweise von links oben nach rechts unten die gesamte Schirmfläche überstreicht (**Bild 83**). Beim Auftreffen auf den Schirm wird die darauf befindliche Schicht zum Leuchten angeregt, wobei man die Helligkeit durch die Strahlintensität beeinflussen kann: Starker Strahl heißt Leuchtpunkt und schwacher Strahl bedeutet Dunkelheit (beim Computer-Bildschirm braucht man keine dazwischenliegenden Helligkeitsstufen zu berücksichtigen). Um ein bestimmtes Zeichen darzustellen, muß nun bloß jemand da sein, der den Strahl bei seinem andauernden Hin und Her im richtigen Augenblick hell und dunkel tastet. Stellen Sie sich diesen Vorgang prinzipiell folgendermaßen vor: Sie nehmen einen feinen Stift und bewegen den (ohne aufzusetzen) wie einen Elektronenstrahl zeilenweise von links oben nach rechts unten über ein Blatt Papier; ab und zu drückt Ihnen jemand kurzzeitig auf die Hand, so daß Sie mit Ihrem Stift ein Pünktchen aufs Papier malen. Dann werden Sie zugeben, daß man auf diese Weise ganze Zeichen zustande bringen und daraus Briefe zusammensetzen kann, wenn man nur konzentriert genug zu Werke geht und im rechten Augenblick am rechten Ort seine Pünktchen macht! Und genau diese Filigranarbeit übernimmt der „intelligente" Steuerbaustein auf Ihrem Video-Interface, der entsprechend wohlklingende Namen trägt: „Steuerbaustein" ist da noch das tiefststapelnde, häufiger finden Sie „Controller" (das heißt trotz millionenfacher Bemühungen nicht „kontrollieren", sondern *„steuern!"*) oder gar „CRT-Controller" (gesprochen „Ssi-Ahr-Tie-Kontroller"), abgeleitet von *Căthode-Ray-Tube-Controller*, was so viel heißt wie „Katodenstrahlröhren-Steuerbaustein".
Insbesondere Neulinge auf diesem Gebiet stehen oft fassungslos vor der Genialität einer solchen Strahlsteuerung (beim Fernsehen ist es noch komplizierter, weil da noch Bewegung und sogar Farbe hinzukommen!); zu Ihrer Erleichterung sei aber noch folgendes ergänzt: Der Strahl wird bei seinem endlosen Weg über den Bildschirm sich nicht selbst überlassen, sondern die ansteuernde Schaltung greift ihm mit zwei ganz wesentlichen Hinweisen unter die Ar-

Bild 84: Bei einer Schreibbreite von 80 Zeichen pro Zeile stimmen Bildinhalt und Ausdruck eines Standard-Druckers genau überein.

Bei 80 Zeichen pro Zeile sind Bildschirm- und Schreibbreite gerade gleich groß.-

Bei 80 Zeichen pro Zeile sind Bildschirm- und Schreibbreite gerade gleich groß._

me: Das ist erstens der Hinweis (sprich: Impuls), eine neue Zeile zu beginnen (horizontale Synchronisation HS), und zweitens ergeht ein separater Impuls, wenn der Strahl links oben mit einem neuen Durchlauf beginnen soll (vertikale Synchronisation VS). Mit diesem zeitlichen Raster ist es ungleich leichter, bestimmte Stellen auf dem Schirm zu treffen, als es bei einem vollkommen freien Strahllauf möglich wäre. Hinsichtlich der Bildwechselfrequenz hat man sich übrigens darauf geeinigt, sie an die Netzfrequenz von 50 Hz anzupassen; pro Bild (und VS-Intervall) bleiben demnach genau 20 ms Zeit, um sämtliche Pünktchen zu aktivieren, und dieser Wert reicht beim Fernseher gerade aus, ein halbwegs flimmerfreies Bild zu erzielen.

Am Netz hängen sie alle

Die Netzfrequenz zieht man deshalb heran, weil dies beim Fernsehen ein besonders einfacher Weg ist, den Strahl-Gleichlauf (Synchronisation) zwischen Sender und Empfänger herzustellen. Und aus demselben Grund der einfachen Reproduzierbarkeit wählt man auch beim Computer-Bildschirm-Betrieb eine Bildwechselfrequenz von 50 Hz entsprechend 20 ms, und für einen Zeilendurchlauf spendiert man (genau wie beim Fernsehen) 64 µs (HS-Intervall); Eingeweihte sollten noch zur Kenntnis nehmen, daß die Ansteuerung bei diesem Video-Interface kontinuierlich, also nicht im Zeilensprung-Verfahren, erfolgt. Da in die genannten Zeiten auch der Strahlrücklauf fällt (vom rechten zurück zum linken Bildrand bzw. von rechts unten zurück nach links oben), reduziert sich der für die Darstellung nutzbare Bereich auf rund 70 % der genannten Zeiten. Sie können sich nun ausrechnen, was Sie in dieser Zeit von knapp 45 µs mit dem Strahl tun müssen, damit der gezielt Pünktchen malt: Bei 80 Zeichen pro Zeile und einer Zeichenbreite von acht Pünktchen bleiben pro Punkt rund 70 ns Zeit (45 µs geteilt durch 8 × 80 Punkte); daraus folgt unmittelbar eine Punktwechselfrequenz von gut 14 MHz, was nicht gerade langsam ist. Wir rechnen Ihnen das deshalb vor, weil es beim Nachbau einer solchen Schaltung auf besondere Sorgfalt ankommt; denn Impulse von 70 ns Dauer sind normalerweise Störspitzen, die ein billiger Oszillograf gar nicht darstellt. Hier aber ist es das Nutzsignal, aus dem ein Pünktchen auf dem Bildschirm wird. Umgekehrt sorgen natürlich Spikes von ein paar Nanosekunden Dauer ebenfalls für Schlieren auf dem Schirm, und daher die gebotene Sorgfalt bei Konzeption und Nachbau, um den Schirm von „Sauereien" freizuhalten.
Für die Schaltungsauslegung entscheidend, das haben Sie eben gesehen, ist die Wahl des Darstellungsformats: Eine Schreibbreite von 80 Zeichen pro Zeile sollte maximal schon möglich sein, weil das ein vernünftiger Kompromiß ist zwischen Bildschirm-Auflösung, Drucker-Schreibbreite und der vertrauten Schreibmaschine (**Bild 84**). Und daß ein Zeichen gerade acht Punkte breit ist, liegt wegen der 8-Bit-Wortlänge unseres Computers und dessen Speicher nahe. Vertikal gehen wir auf 24 (Text-) Zeilen, weil das ein ästhetisches Schriftbild ergibt und auch marktüblich ist. Für die Zeichendarstellung haben wir beim MOPPEL eine Matrix von 8 × 12 Punkten gewählt, was einen Kompromiß zwischen guter Lesbarkeit, Speicherplatz-Ausnutzung und den verschiedenen möglichen Bildformaten darstellt. Verschiedene Formate deshalb, weil ein Fernseher (und das Mini-Sichtgerät) keine 80 × 24 Zeichen auf dem Schirm unterbringt; der Fernseher wegen der mangelnden Bandbreite seiner internen Verstärker und das Mini-Sichtgerät wegen des kleinen Bildschirms, der noch ohne Lupe ablesbar sein sollte. Daher haben wir das Video-Interface umschaltbar gestaltet, so daß Sie jederzeit (über zwei kleine Brücken) bestimmen können, ob Sie auf Ihrem Schirm Breitwand (24 × 80 Zeichen) oder Schmalfilm (18 × 40 Zeichen) sehen wollen; an der Zeichendarstellung in Form einer 8 × 12-Punkt-Matrix ändert sich dadurch nichts.

Hektische Betriebsamkeit

Es ist recht kurzweilig, sich das Treiben auf dem Video-Interface einmal genauer anzusehen. Betrachten Sie dazu einmal die Blockschaltung (**Bild 85**) und das **Bild 86**, das die linke obere Ecke des Bildschirms mit der Bereitmeldung Ihres Computers zeigt; im

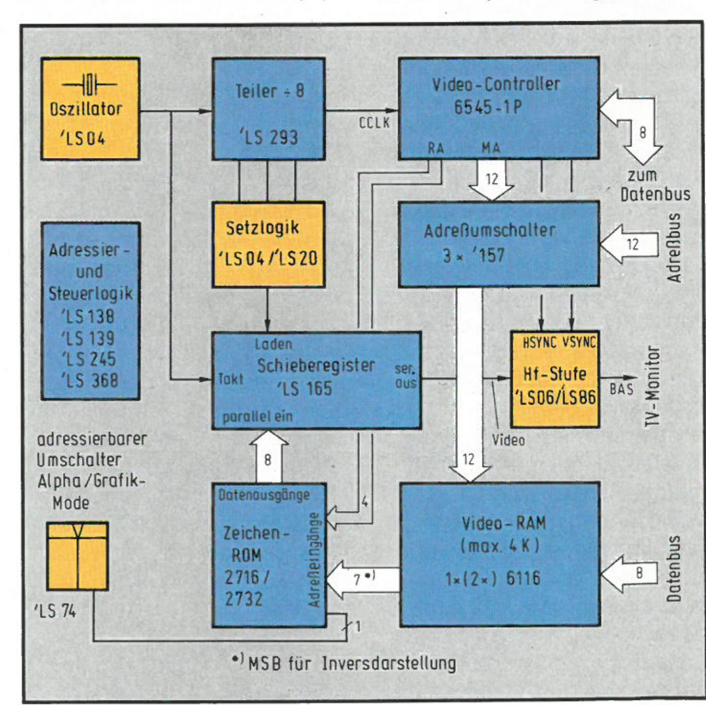


Bild 85: Blockschaltung des Video-Interfaces.

oberen Schirmausschnitt sehen Sie die ersten sieben *Text*zeilen, und in der Bildmitte als vergrößertem Ausschnitt die erste (sowie Ansätze der zweiten) Textzeile. Durch die Vergrößerung erkennen Sie hier bereits die zwölf *Bild*zeilen, aus denen sich jede Textzeile zusammensetzt (unter „Bildzeile" ist *ein* Strahldurchlauf zu verstehen); außerdem erkennen Sie das Pünktchenmuster, das die einzelnen Buchstaben ergibt. Im Bild unten schließlich ist der zeitliche Verlauf des Strahlstroms dargestellt, wie er bei Durchlaufen der vierten Bildzeile zu beobachten ist: HIGH-Pegel bedeutet immer ein leuchtendes Pünktchen, während der Strahl bei LOW-Ansteuerung dunkel bleibt. Das Pünktchenmuster findet der Steuerbaustein in einem EPROM, das pro Zeichen 12 Bytes gespeichert hat; wie Sie richtig bemerken, ist das Erstellen eines solchen Zeichengenerators eine Sklavenarbeit, die hinterher niemand mehr würdigt.
Nehmen wir einmal an, der Steuerbaustein beschließt, eine neue Bilddarstellung zu beginnen. Dann gibt er dazu die Adresse MA0...MA11 (*Memory Address*) für die erste Speicherzelle im Bildwiederholspeicher (Video-RAM) aus, und zwar zwölfstellig, wie es sich für einen 4 K großen Speicher gehört (auch wenn davon nur 2 K bestückt sind). Das RAM reagiert, wie wir es erwarten: Es gibt an seinen Daten-Ausgängen diejenige Information aus, die es unter der aktivierten Adresse augenblicklich gespeichert hat; in unserem Beispiel ist das der ASCII-Code für das „M", der nach alter Väter Sitte „4D" lautet. Diese Information gelangt als oberer Teil der Adresse 4DX ans Zeichen-PROM; der untere Teil „X" stammt direkt vom Steuerbaustein, der an seinen Ausgängen RA0...RA3 (Row Address) angibt, welche der zwölf Bildzeilen er gerade bearbeitet (im Beispiel ist es die vierte Bildzeile, entsprechend binär „3", da die Zählung mit 0 beginnt). Und nun hat das ROM keine andere Wahl mehr, als an seinen Datenausgängen das unter der Adresse 4D3 abgelegte achtstellige Bitmuster herauszurücken, das dem Strahl angibt, in welcher Aufeinanderfolge er hell und dunkel sein soll (im Beispiel lautet das Bitmuster für die vierte Pünktchenzeile des „M" gerade 01011010=5A; **Bild 87**). Wie Sie aus der Darstellung ersehen, sind im Zeichen-EPROM immer vier Bytes bis zum nächsten Zeichen frei (hier von 4DC...4DF); das verschwendet zwar 25 % Platz, hat aber den Vorteil der klaren Zuordnung zwischen ASCII-Code und zugehörigem Punktmuster (das „a" mit dem ASCII-Code „61" finden Sie z. B . sofort ab Adresse 610 im Zeichen-EPROM).
Mit diesem 8-Bit-Wort aus dem Zeichen-EPROM wird das Schieberegister geladen, an dessen Ausgang die Bits kleckerweise herausfallen, um den Strahl je nach ihrem Pegel hell oder dunkel zu tasten. Nicht träge, verlangt der Steuerbaustein nach jeweils acht Bits vom *nächsten* Zeichen das Muster für die entsprechende Bildzeile; dazu muß er die Adresse MA um Eins erhöhen (um die nächste RAM-Zelle anzusprechen), während die Information an RA konstant bleibt (erst muß die gesamte Bildzeile ja durchlaufen werden). Am Ende einer Bildzeile wird RA dann um Eins erhöht, während MA wieder auf den Anfang der Textzeile zurückgeht. Die Hardware-Organisation sieht übrigens so aus, daß die Speicherzellen im Video-RAM fortlaufend den Plätzen auf dem Schirm zugeordnet sind, beginnend links oben. Bei 24 × 80 = 1920 Zeichen (= 780 HEX) landet die Information aus Adresse 77F (= 780-1) folglich rechts unten auf dem Bildschirm.
Wer das alles zum ersten Mal liest, hält es für ein Wunder, weil es in der Praxis tatsächlich funktioniert. In Wirklichkeit ist so ein Steuerbaustein nichts anderes als ein Zählerbergwerk, bei dem Sie für die diversen Signale die maximalen Zählerstände vorgeben können, und das Sortieren und Ausgeben besorgt dann

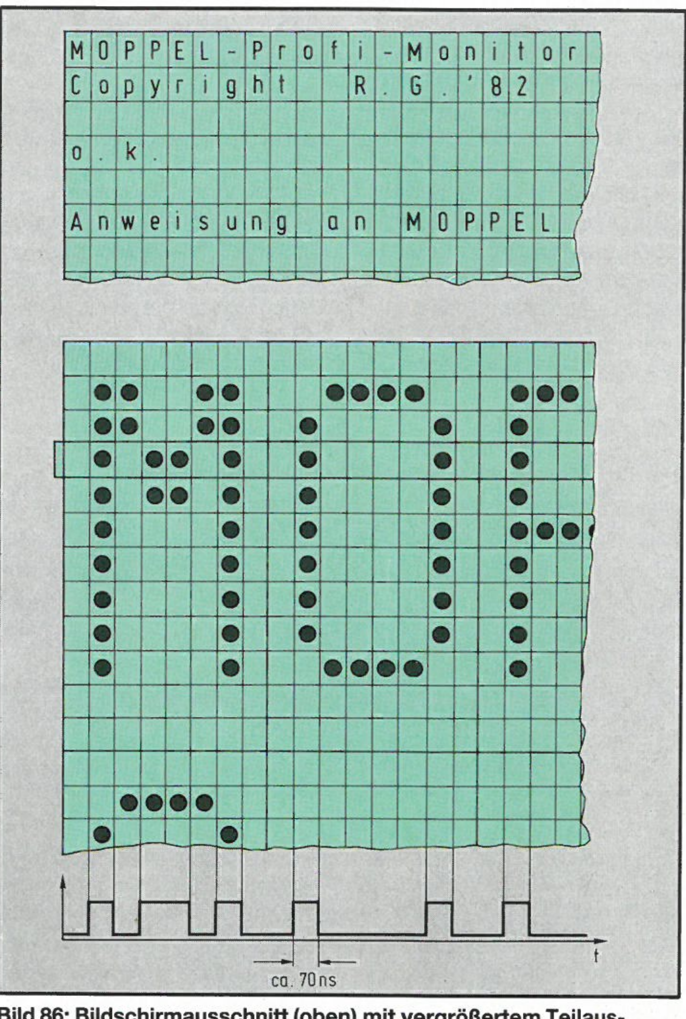


Bild 86: Bildschirmausschnitt (oben) mit vergrößertem Teilausschnitt (Mitte) und Strahlstromverlauf während der vierten Bildzeile (unten).

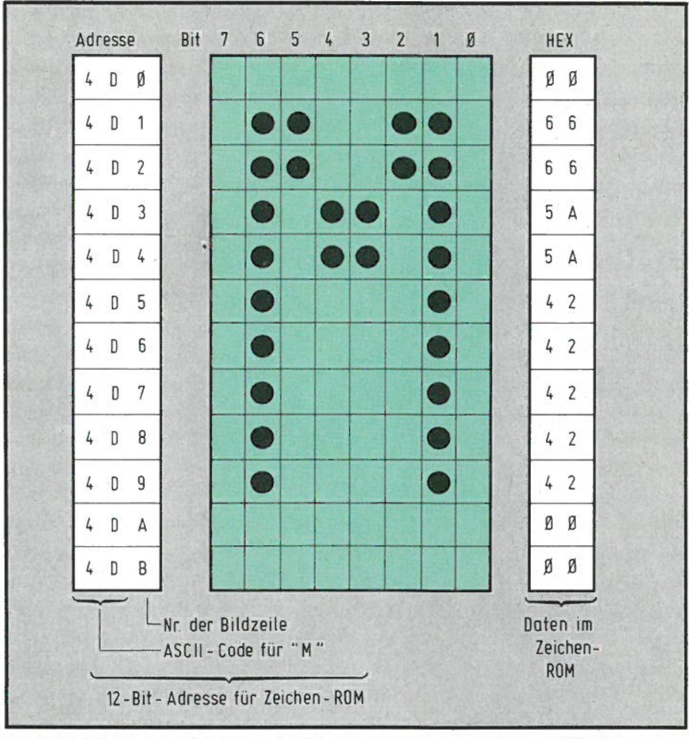


Bild 87: Für jedes Zeichen sind im Zeichengenerator zwölf Bytes reserviert, deren HIGH-Bits bei der Darstellung auf dem Schirm zu Leuchtpunkten werden.

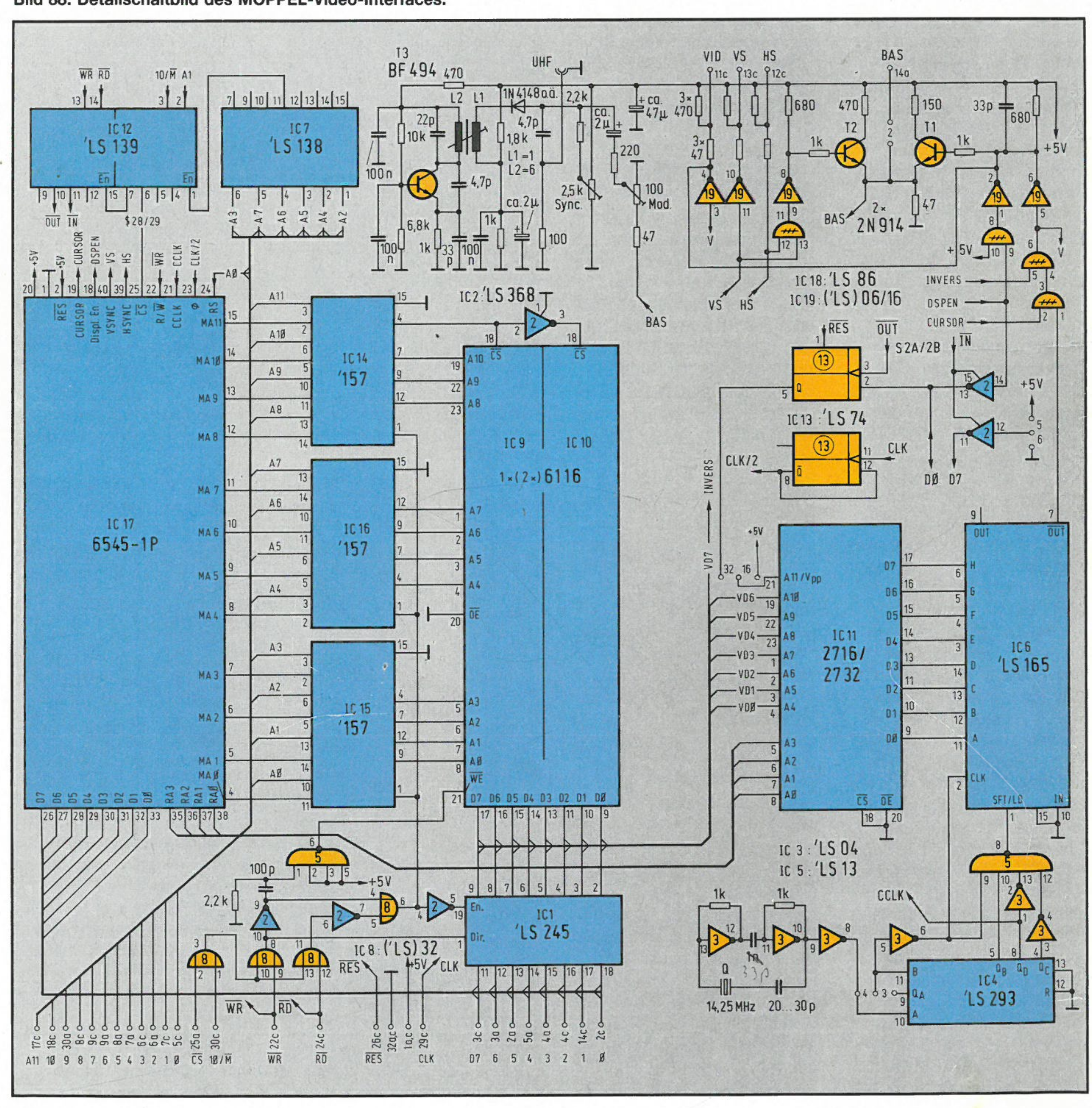
das IC; dafür ist es schließlich 40beinig, teuer genug und auch nicht gerade zimperlich mit der Stromaufnahme! Dennoch lassen Sie sich am Rande gesagt sein, daß der Entwurf und die Umsetzung einer solchen Interface-Schaltung schon einiges an Kopfzerbrechen erfordert; und wenn Sie mögen, dann erfahren Sie auch noch den Rest der Wahrheit über diese Baugruppe.

Schwarzsehen verpönt

An ASCII-Zeichen gibt es ganze 128 Stück, zu deren Unterscheidung sieben Bits nötig sind; im Video-RAM aber belegt jedes ASCII-Zeichen eine Speicherzelle mit acht Bit Wortlänge – was also wird aus dem freien achten Bit (dem höchstwertigen)? Wie Sie dem Detailschaltbild (**Bild 88**) entnehmen, gelangt dieses Bit nicht als Teil der Adresse ans Zeichen-ROM, sondern geht als INVERS-Bit mit in die Bildsignal-Aufbereitung ein: HIGH-Pegel im höchstwertigen Bit eines ASCII-Zeichens invertiert (über ein Exklusiv-ODER-Gatter) die vom Zeichen-ROM stammenden Daten, so daß die Schrift nicht mehr auf dunklem, sondern erhelltem Hintergrund erscheint; so einfach geht das beim Fernseher!

Quasi in Klammern fügen wir hier noch eine Feinheit ein, auf die Sie spätestens dann stoßen, wenn Sie in die Schaltungstiefen dieser Karte vordringen: Wenn der Steuerbaustein eine MA-Adresse ausgibt, dann dauert es eine Weile, ehe die dazupassenden Daten aus dem Zeichen-EPROM ins Schieberegister

Bild 88: Detailschaltbild des MOPPEL-Video-Interfaces.



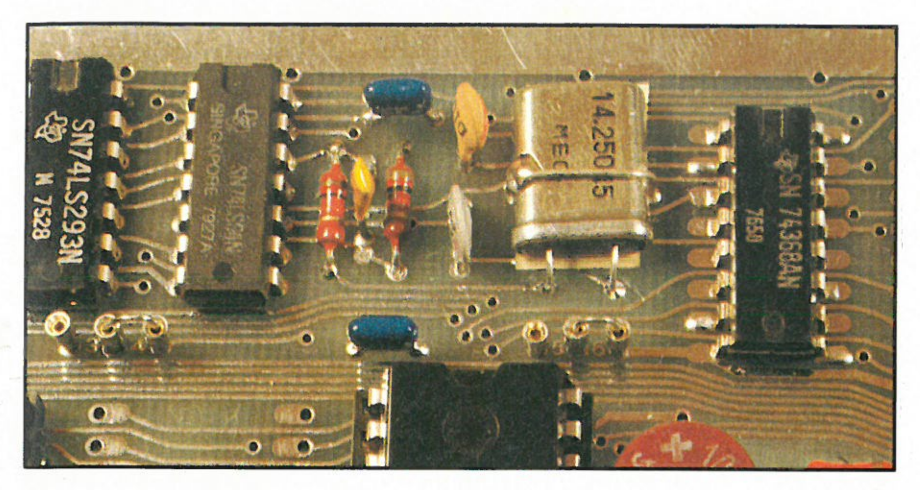


Bild 89: Nach der Montage auf einer Isolierscheibe soll das Quarzgehäuse mit dem Haltebügel verlötet werden.

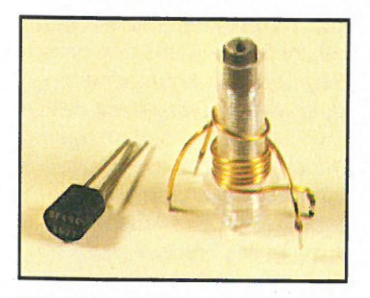


Bild 90: Lediglich sieben Windungen sind zum Bewickeln der Oszillatorspule aufzubringen.

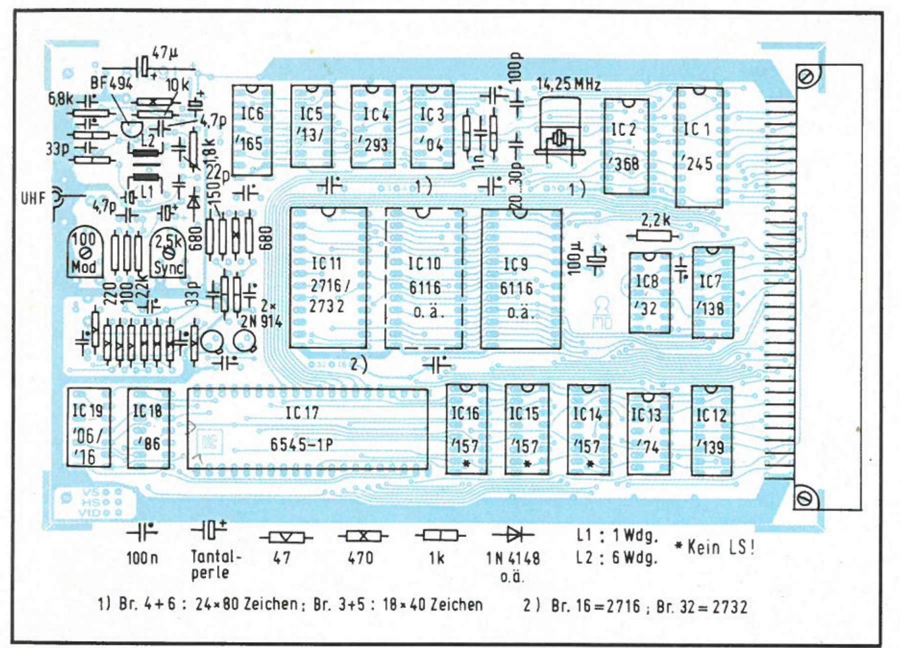


Bild 91: Bestückungsplan; der UHF-Modulator links oben wird nur dann bestückt, wenn ein normales Fernsehgerät angeschlossen werden soll.

gelangt sind (Laufzeiten, Speicherzugriffszeiten, die bei diesen Geschwindigkeiten nicht mehr vernachlässigbar sind). Auf dem Schirm erscheint daher (durch einen Trick im Steuerbaustein) nicht das augenblicklich adressierte Zeichen, sondern das davor angesprochene (Verzögerung von DSPEN um ein Zeichen; s. u.).

Wenn Sie aufmerksam mitgezählt haben, sind Sie beim Zeichen-ROM auf insgesamt elf Adreßbits gekommen (sieben stammen vom ASCII-Zeichen aus dem Video-RAM und vier von den RA-Ausgängen des Steuerbausteins). Mit elf Bits lassen sich bekanntlich 2 K Speicher adressieren, für IC 11 aber kann auch ein 2732-EPROM mit 4 KBytes Umfang eingesetzt werden – woher stammt das dafür erforderliche zwölfte Bit? Das kommt geradewegs vom Ausgang eines Flipflops, welches über den Befehl „OUT 2A" ansprechbar ist. Wird zusammen mit diesem Ausgabebefehl aus dem niedrigstwertigen Bit des Akkumulators HIGH-Potential übertragen, geht am Zeichen-ROM (sofern es ein 2732 ist) das Adreßbit 11 auf HIGH und schaltet den gesamten Zeichengenerator auf die obere 2-K-Hälfte um, wo die Grafikzeichen angesiedelt sind. Zurück in den alphanumerischen Betrieb geht es mit Gewalt über RESET oder dezent über die Ausgabe von „00" an die Portadresse 2A (ans Flipflop). Entsprechend dem verwendeten Zeichen-EPROM ist die Brücke „16" bzw. „32" einzusetzen.

Die im Schaltplan (und auf der Platine) noch herumhängenden Tri-State-Ausgänge in IC 2 landen nach einem IN-2A-Befehl im Akkumulator. Bit 7 gibt dabei den Zustand des DSPEN-Signals an (Display Enable); dieses Signal stammt vom Steuerbaustein und es besagt, daß sich der Strahl gerade im aktiven Teil seiner Laufbahn befindet (im Gegensatz zum Strahlrücklauf, wo dieser dunkel getastet wird). Bit 0 gibt den durch die Brücken 5 bzw. 6 eingestellten Pegel an die CPU weiter; die erkennt daran, welches Bildformat Sie gerne hätten (24 × 80 Zeichen = Br. 6 oder 18 × 40 Zeichen = Br. 5) und teilt dieses Begehren dem Steuerbaustein mit. Der ist für solche Wünsche offen und könnte ebenso 39 oder 97 Zeichen pro Zeile verwalten, wenn er entsprechend vorbereitet („programmiert") werden würde. Parallel zur Einstellung dieser Brücken müssen Sie noch eine der Brücken 3 oder 4 einsetzen: Mit Br. 4 gelangt die volle Oszillatorfrequenz an die nachfolgende Schaltung, was pro Zeile 640 Takte bedeutet, entsprechend 80 Zeichen; mit Br. 3 wird die Taktfrequenz halbiert, so daß pro Zeile eben nur noch halb so viel Zeichen hineinpassen. Wegen der im Bildschirmgerät konstanten Zeilenfrequenz genügt es hierbei nicht, nur den Steuerbaustein umzuprogrammieren.

Die beiden Transistoren T1 und T2 bilden einen Differenzverstärker, in dem die Zusammensetzung des BAS-Signals (Bild-, Austast- und Synchronsignal) erfolgt. Im Prinzip muß dabei das vom Schieberegister stammende Video-Signal nur noch mit dem horizontalen (Zeilen-) und vertikalen (Bild-) Synchron-Impuls verknüpft werden, was dank der guten Zuarbeit vom Steuerbaustein (der generiert diese Signale, wiederum in alle Richtungen variabel) ziemlich problemlos vor sich geht.

Ein Fernsehsender im Eigenbau

Das BAS-Signal dient zur Einspeisung in den Video-Eingang eines TV-Monitors, während die Einzelsignale VID, HS und VS zur Ansteuerung unseres Mini-Sichtgerätes herausgeführt sind (die Brücke 2 ist auf der Platine bereits realisiert). Wer seine gute alte

Glotze zur Anzeige verwenden will, muß im allgemeinen den Weg über den Antenneneingang wählen. Dazu dient der UHF-Modulator, der auf der Platine ebenfalls noch Platz hat, und dessen Ausgangssignal direkt in den Antenneneingang des Fernsehers einzuspeisen ist. Wegen der zumeist fehlenden Schutzisolation raten wir dringend vom nachträglichen Einbau einer Video-Buchse in einen alten Fernseher ab!
Hinter dem unscheinbaren Begriff „Modulator" verbirgt sich übrigens nichts anderes als ein waschechter Sender, noch dazu einer im ultrahohen Frequenzbereich (wo es doch ab 10 kHz schon strafbar wird!). Aber keine Angst, wenn Sie diesen Sender nur zum direkten Anschluß an den Antenneneingang Ihres Fernsehapparates verwenden, gibt Ihnen die Post dafür eine Blanko-Vollmacht (Amtsblatt Nr. 97 vom 24. 7. 1970: „Allgemeine Genehmigung für den Betrieb von EDV-Anlagen, die zur Art der Tischrechner gehören...").

Wegen der geringen Bandbreite kann zusammen mit einem Fernsehgerät nur das kleine Bildformat von 18 × 40 Zeichen eingestellt werden!

Beim Nachbau sind eigentlich nur ein paar Dinge zu beachten, die aber dafür gründlich: Fassungen sollen nur da eingesetzt werden, wo sie vom Bausatz her vorgesehen sind: für IC 9 (Video-RAM), IC 11 (Zeichen-EPROM) und IC 17 (Steuerbaustein); an anderer Stelle handeln Sie sich mit dem Einsatz von Fassungen möglicherweise größere Probleme ein als irgendeinen Nutzen. IC 10 bleibt unbestückt; dieser Platz ist für Sonder-Anwendungen vorgesehen. Der Quarz ist auf einer Isolierscheibe (z. B. Papier) liegend zu montieren und mit einem Drahtbügel zu fixieren (Bügel mit dem Quarzgehäuse verlöten; **Bild 89**). Sie müssen drei Brücken einlöten: Br. 3 *und* Br. 5 für das kleine Darstellungsformat (18 × 40 Zeichen) bzw. Br. 4 *und* Br. 6 für das volle Format von 24 × 80 Zeichen; mit der dritten Brücke (16 bzw. 32) nehmen Sie die Anpassung an den eingesetzten Zeichengenerator vor. Entsprechend dem Bestückungsplan verstreuen Sie noch ein paar keramische Entstör-Kondensatoren, die übrigens keinen Selbstzweck erfüllen; sie sorgen für Reinheit auf der Versorgungsspannung (d. h. die Unterdrückung von Störspitzen), die bei der erwähnten Sensibilität der Schaltung oberstes Gebot ist.
Beim UHF-Modulator müssen Sie die Spule selbst wickeln, besser gesagt das Spülchen; denn 1 + 6 Windungen sind sicherlich nicht der Rede wert (sechs Windungen im Schwingkreis am Transistor und eine zum Auskoppeln auf der anderen Seite). Sie sollten den Spulenkörper vor dem Einsetzen bewickeln, die Drahtenden verzinnen, und dann die ganze Einheit (richtig herum!) einlöten (Körper bei Überlänge evtl. kürzen; **Bild 90**). Bei halb eingedrehtem Kern und angeschlossenem Fernseher stellen Sie einen freien Kanal ein und verdrehen den Kern so weit, bis die Schrift der MOPPEL-Meldung auf dem Schirm auftaucht, auch wenn sie zuerst noch „durchläuft"; mit dem Sync-Poti stellen Sie den Synchronisationspegel für den „Bildfang" ein, und das Mod-Poti bestimmt den Modulationsgrad (sprich: Bildhelligkeit); diese Einstellarbeit ist ein Klacks (mundartlich für „Kleinigkeit") für jemanden, der vorher gerade ein komplettes Video-Interface zusammengelötet hat (**Bild 91**).
Die Bildqualität auf dem Fernsehschirm ist nicht gerade überragend, weil so ein Ding eben für Krimskrams ausgelegt ist und nicht für Computer-Daten (**Bild 92**); es lohnt sich daher sicher, auf einen (bereits ab ca. 300,– DM verfügbaren) TV-Monitor zu sparen!
Beim späteren Betrieb Ihres MOPPEL steht das Video-Interface vollkommen im Hintergrund; vielleicht denken Sie hin und wieder mal dran, daß sich auf dieser Karte mindestens ebensoviel tut wie auf der Zentraleinheit, nur bringt es eine solche Interface-Platine eben nie auf die Popularität eines Mikrocomputers, selbst wenn sie siebenmal so schnell arbeitet wie dieser!

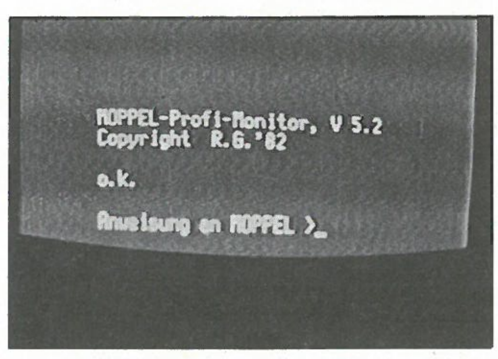


Bild 92: Durch den Umweg über den Antenneneingang und die begrenzte Bandbreite ist die Bildqualität auf dem Fernseh-Bildschirm deutlich schlechter als auf einem TV-Monitor.

Weitblickende Software

Mit Hilfe des Video-Monitors im gelben EPROM unterhalten Sie sich mit Ihrem MOPPEL über Bildschirm und ASCII-Tastatur.
Das wilde Zusammenlöten eines Mikroprozessors mit einer Tastatur und einem Video-Interface ergibt einen Haufen Hardware, aber noch lange keinen Mikrocomputer; was dem zu einem erquicklichen Erdendasein fehlt, ist (außer dem planvollen Zusammenbau) ein Gehirn, sprich: ein fest im EPROM abgespeichertes Betriebsprogramm. Bei ganz kleinen Systemen, die man gerade in Maschinensprache programmieren kann, nennt man diese minimale Intelligenz zur Systemverwaltung „Monitor" oder auch „Monitorprogramm", was auf keinen Fall zu verwechseln ist mit einem Datensichtgerät, das hin und wieder auch auf den Namen „Monitor" hört. Bei größeren Computern heißen die Software-Pakete zur Eigenverwaltung „Betriebssystem", was vom Namen her schon den Hauch des Unnahbaren in sich birgt. Die Betriebssoftware für den bildschirmorientierten MOPPEL bewegt sich zwischen den beiden genannten Ebenen, ohne daß wir dafür eine Benennung kreieren wollen; Sie wissen (oder erfahren es jetzt), daß für den Bildschirmbetrieb des MOPPEL die beiden EPROMs rot *und* gelb (Profi-Monitor und dessen Erweiterung, der Video-Monitor) erforderlich sind; die Monitor-Versionen grün und blau (Null- bzw. Basismonitor) sind ausschließlich für den Betrieb mit der HEX-Tastatur vorgesehen. Um bezüglich der verschiedenen Monitor-EPROMs alle Unklarheiten zu beseitigen, betrachten Sie bitte einmal **Bild 93**: Auf dem Steckplatz #0 der CPU befindet sich der (rote) Profi-Monitor im 2732-EPROM, das den Adreßbereich ab 0000 belegt; nach jedem Rücksetzen (und nach dem Einschalten) beginnt MOPPEL seine Arbeit folglich mit dem in diesem EPROM abgespeicherten Programm. Dort steht ziemlich zu Anfang die intelligente Frage, die der Mops stumpfsinnig nach je-

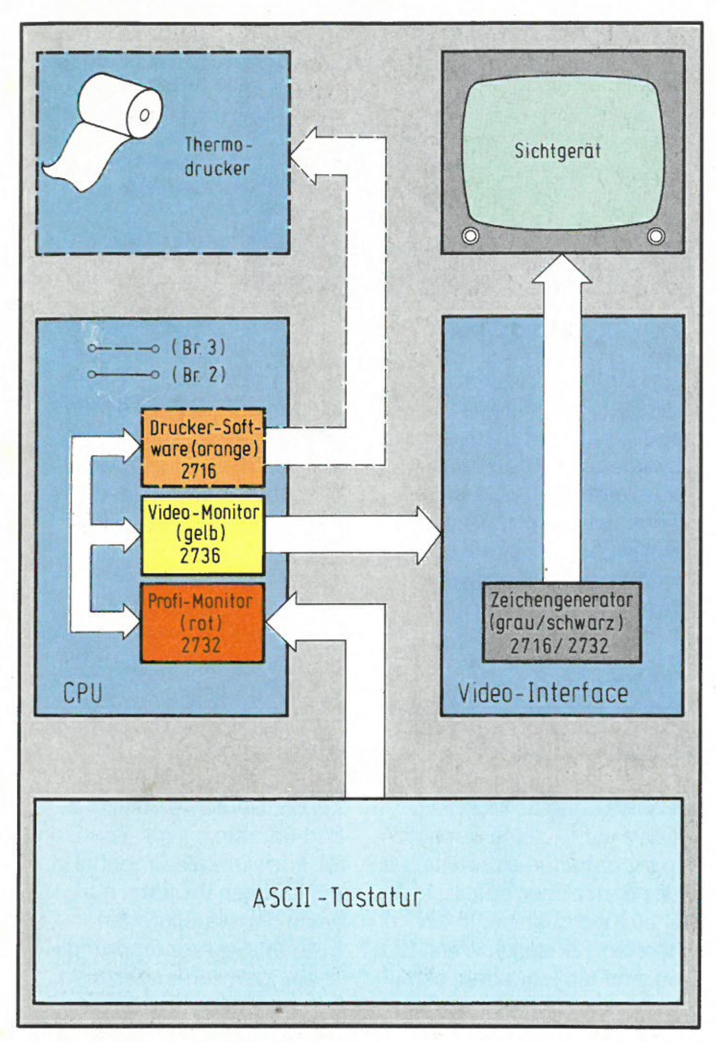


Bild 93: Mit den verschiedenen EPROMs läßt sich die Leistungsfähigkeit des Systems der jeweiligen Ausbaustufe anpassen.

dem RESET immer wieder neu stellt: Welche Tastatur ist eigentlich angeschlossen, die hexadezimale oder die „richtige" Schreibmaschinentastatur? Im ersten Fall bewegt sich der Prozessor weiter im EPROM auf Platz #0, wissend, daß sein Bediener (über die HEX-Tastatur) in Maschinensprache mit ihm zu reden wünscht. Ist dagegen die ASCII-Tastatur angeschlossen, erfolgt zielstrebig ein Sprung ins „gelbe" EPROM (ab Adresse 1000), wo derjenige Teil des µP-Verstandes abgelegt ist, der zum Betrieb des bildschirmorientierten Systems erforderlich ist. Rotes und gelbes EPROM gehören weder zum Lieferumfang der CPU noch zum Video-Interface; sie sind als Zubehör gesondert zu beziehen.

Der feine Unterschied

Um den Unterschied zwischen beiden Tastaturen herauszubekommen, liest MOPPEL die Tasten-Zeile „BC" ein, die es auf beiden Tastaturen gibt; nur auf der HEX-Tastatur liegt dabei ein winziges Bit auf LOW (verursacht durch die Germanium-Diode), während bei der ASCII-Tastatur (im Ruhezustand) acht HIGH-Bits eingelesen werden; und an diesem feinen Unterschied eines LOW-Bits, der nicht einmal 1 µs lang dauert, erkennt MOPPEL den Unterschied zwischen HEX- und ASCII-Eingabe.
Das dritte EPROM auf Platz #2 der CPU-Platine ist für die Software (und den Zeichengenerator) des Thermodruckers vorgesehen; es ist im Lieferumfang dieser Baugruppe enthalten. Natürlich wird dieses orange EPROM nur dann benötigt, wenn der Thermodrucker im System vorhanden ist. Zum Betrieb eines großen Druckers (externes Peripheriegerät) ist das orange EPROM *nicht erforderlich.*
Von den drei eben genannten EPROMs vollkommen unabhängig arbeitet der Zeichengenerator auf dem Video-Interface; in diesem grauen (bzw. schwarzen) EPROM ist auch kein Programmteil enthalten, sondern hier stehen, in Form einer umfangreichen Tabelle, diejenigen Bitmuster, die bei entsprechender Ansteuerung die Zeichen auf dem Bildschirm ergeben.
Auch dieser Zeichengenerator ist in der (grauen) Standard- bzw. der (schwarzen) Erweiterungs-Version Bestandteil des Video-Interfaces.
Nachdem Sie wissen, welches EPROM wofür zuständig ist, können Sie Ihren MOPPEL als Partner betrachten, von dem Sie nur noch gewisse Eigenschaften interessieren; daß man dieses Verhalten per Programm züchten, d. h. vorgeben und beeinflussen kann, versteht sich bei so einer Maschine von selbst. Der Umgang mit einem Computer entbehrt nicht einer gewissen Sturheit, die manchmal so weit geht, daß sie dem einen oder anderen unheimlich vorkommt. Rechnen Sie ihm (Ihrem Computer) dieses hölzerne Verhalten bitte nie als Überheblichkeit an, sondern eher als Doofheit, die zu mehr eben nicht reicht. Das geht im Einzelfall sogar so weit, daß sich der Blödmann vor Ihnen über ein fehlendes Komma aufregt, das man als denkender Mensch im Schlaf einfügen könnte; aber denken kann er nicht, Ihr Computer, und wie alle Doofen muß er daher pedantisch auf winzigsten Kleinigkeiten bestehen. Wenn Sie das berücksichtigen, sind Sie später nicht ganz so sauer, wenn am Anfang Ihrer Beziehungen permanent das Gemecker über Bedienungsfehler ertönt (**Bild 94**); nach einer kurzen Gewöhnungsphase geht Ihnen der Dialog mit Ihrem Gegenüber ohnehin in Fleisch und Blut über, und Sie verstehen gewisse Verhaltensweisen ganz selbstverständlich als Charaktereigenschaften.

Der Ton macht die Musik

Gehen Sie mit dieser Einstellung bitte an Ihren MOPPEL heran, wenn Sie sich an den richtigen Umgangston herantasten, und bewegen Sie dabei die folgenden Worte in Ihrem Herzen: MOPPEL ist kein

Tabelle 7:
Funktionen des Video-Monitors in alphabetischer Reihenfolge

- A – Assembler-Aufruf
- B – Basic-Aufruf
- C – COPY (Speicherbereich kopieren)
- D – Disketten-Betriebssystem aufrufen
- E – Text-Editor aufrufen
- F – FILL (RAM mit Konstante füllen)
- G – GO (Programmstart)
- H – HEX-Dump (Speicherinhalt ausdrucken) (Thermodrucker)
- I – Kassetten-Input
- J – Bildschirm löschen
- K – Klartext ausdrucken (Thermodrucker)
- L – LIST (Speicherinhalt darstellen)
- M – Programmieren in Maschinensprache
- N – Bildschirminhalt ausdrucken
- O – Kassetten-Output
- P – EPROM-Programmierzusatz aktivieren
- Q – Bildschirminhalt zurückholen
- R – Bildschirminhalt retten
- S – Single Step (Einzelschritt-Betrieb)
- T – Text-Mode
- U – Echtzeit-Uhr lesen/stellen
- V – Serielle Ausgabe
- W – Serielle Eingabe
- X – Umkopieren mit Adreßanpassung
- Y – Einzelbyte-Betrieb
- Z – Breakpoint-Betrieb

BASIC-Computer, aber er kann natürlich auch BASIC; MOPPEL ist auch kein Entwicklungssystem, obwohl Sie ihn selbstverständlich dafür einsetzen können; und ohne Frage läßt MOPPEL auch die Programmierung in Maschinensprache zu, aber eben nicht nur ausschließlich. Verstehen Sie diesen Computer so, daß er eine zentrale Verstandesebene hat (Anweisungs-Ebene), die auf ein Stichwort (genauer: Stichbuchstaben; **Tabelle 7**) von Ihnen reagiert, und die daraufhin auf andere Ebenen verzweigt, stets aber auf diese eine Basisstufe zurückkehrt. Und diese Ebene der Verständigung erkennen Sie daran, daß Ihr Mops in Lauerstellung geht und Sie permanent auffordert: „Anweisung an MOPPEL >" zu geben (**Bild 95**). Hier geht dann die eben angesprochene Pingeligkeit los, denn an Anweisungen akzeptiert MOPPEL jeweils nur einen einzelnen Buchstaben, dafür aber 26 verschiedene (weil sich unser Alphabet auf diese Anzahl beschränkt); es ist MOPPEL völlig egal, ob Sie Eingaben in Groß- oder Kleinschrift vornehmen, nur Buchstaben müssen es halt sein. Und wenn Sie meinen, von dem vorgeschriebenen Format abweichen zu müssen, beginnt die ebenfalls schon angesprochene Meckerei: „Diese Anweisung kenne ich nicht" (**Bild 96**).

Wie auch bei anderen sturen Zeitgenossen, müssen Sie einen Computer jedesmal treten, bevor er mit seiner Arbeit loslegt; das ist gewissermaßen ein Klaps auf den Hintern, der ihm sagt, daß *Sie* fertig sind mit Anweisungen und *er* dran ist mit Arbeiten. Diese Übergabe der Aktivitäten nehmen Sie per Return-Taste vor (das ist die winkelige ganz rechts unten). Zwischen dem Kennbuchstaben für die gewünschte Aktivität und dem Druck auf Return müssen Sie MOPPEL noch sagen, *wo* er im einzelnen tätig werden soll; d. h. Sie müssen (soweit erforderlich) Anfangs-, End- und Zieladressen für die betreffende Operation angeben (Buchstaben und folgende Adressen stets durch ein Leerzeichen trennen!). Fachleute nennen dies „Randbedingungen" oder „Parameter" vorgeben. Wie das im einzelnen abläuft, sehen wir uns gleich an, zuvor machen Sie sich bitte noch mit folgenden Verhaltensweisen vertraut:

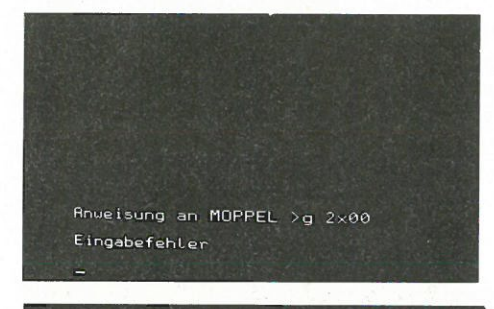


Bild 94: Wenn eine Bediener-Eingabe vom vorgeschriebenen Format abweicht, beschwert sich der Computer darüber.

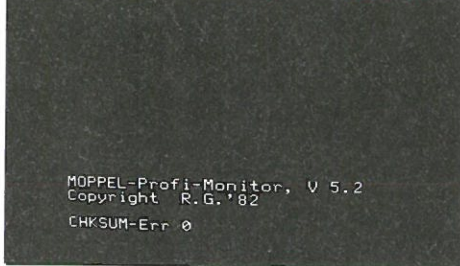


Bild 95: Nach dem Einschalten zeigt die Bereitmeldung an, daß das System Anweisungen vom Bediener entgegennimmt.



Bild 96: Als Anweisung sind nur die Buchstaben unseres Alphabets zugelassen, wahlweise in Klein- oder Großschreibung.

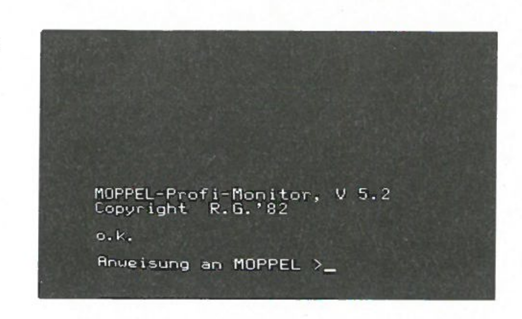


Bild 97: Bei fehlerhaftem EPROM-Inhalt verweigert MOPPEL die Arbeit, um falsche Reaktionen zu verhindern.

Augenzwinkernder Platzanweiser

Als Anweisung an MOPPEL werden nur einzelne Buchstaben akzeptiert, die unmittelbar hinter dem Hinweispfeil „>" stehen (und auch nur in der untersten Textzeile). Wenn Sie dran sind mit Eingaben, signalisiert Ihnen MOPPEL das mit einem permanenten Augenzwinkern, d. h. einem blinkenden *Cursor* (sprich: Körßer); bei ausgeschaltetem Cursor erkennen Sie sofort, daß die Maschine vor Ihnen in irgendeiner Weise beschäftigt ist und keine Eingaben von Ihnen entgegennimmt. Vornehme Leute nennen den Cursor übrigens „Leuchtzeiger", was genauso treffsicher übersetzt ist wie „Weichware" für „Software"; der Cursor ist im Prinzip nichts anderes als ein Platzanweiser, der Ihnen und der Tastatur sagt: Das nächste Zeichen, das ankommt, gelangt an die Stelle, an der ich gerade blinke. Und nach einer jeden Eingabe wandert dieser freundliche Blinker eine Stelle nach rechts, weil er davon ausgeht, daß Sie als wohlerzogener Mitteleuropäer von links nach rechts schreiben. Diesen Cursor haben wir beim MOPPEL klein und dezent gestaltet (3 × 8 langsam blinkende Bildpunkte), damit Ihnen beim Blick auf Ihren Bildschirm nicht die aggressive Penetranz entgegenstrahlt, die einem riesengroßen, irrsinnig schnell blinkenden Cursor anhaftet (der die Mitschuld für viele Computer-Verteufelungen trägt!).

Ferner nimmt es Ihnen das Monitor-Programm ab, bei Adressen führende Nullen eingeben zu müssen: Statt „004D" brauchen Sie folglich nur „4D" einzugeben. Und ergänzend können Sie noch wahrnehmen, daß das „o. k." bei der Bereitmeldung (vgl. Bild 95) kein billiger Schnörkel ist, sondern das Ergebnis einer internen Prüfung; nur wenn die mitgelieferten EPROMs in ihrer ursprünglichen Form vorhanden sind, kommt dieses „o. k.", und MOPPEL macht willig weiter. Sollten Sie an den Monitor-EPROMs herumgewurschtelt haben, quittiert MOPPEL dies mit einem störrischen „CHKSUM-Err" (**Bild 97**), der das ganze System blockiert. Die 26 in Tabelle 7 zusammengestellten MOPPEL-Anweisungen lassen sich inhaltlich in sieben Gruppen einteilen (**Tabelle 8**).

1. Die Anweisungen auf Maschinenebene beziehen sich auf die Programmierung in Maschinensprache, einschließlich verschiedener Speicher-Operationen wie Listen, Kopieren, Verschieben oder Auffüllen von Speicher-Bereichen. Als Unterstützung der Programmierung auf dieser maschinennahen Ebene dient Ihnen der 8085-Befehlssatz auf zwei Postern (in ELO 3 und 4/83) sowie die Erläuterung dieser Befehle anhand von Beispielen im **ELO-Labor-Brief Nr. 1.**
2. Neben den direkten (Pfeil-) Tasten zur Cursor-Steuerung

Tabelle 8:
Kommandos des Video-Monitors, nach Funktionsgruppen geordnet

1. Anweisungen auf Maschinen-Ebene

M – Programmieren in Maschinensprache; Speicherinhalt inspizieren und modifizieren
G – Programmstart von der angegebenen Adresse aus
L – Speicherinhalt (hexadezimal) auf dem Bildschirm darstellen
F – RAM-Bereich mit Konstante auffüllen
C – Speicherbereich kopieren
X – Programmteile verschieben (einschließlich Adreßanpassung)

2. Bildschirm-Kommandos:

J – Bildschirm löschen
R – Inhalt des Video-Speichers (ab 3000) in den Arbeitsspeicher (ab 8000) überschreiben („retten")
Q – Video-Speicher mit den zuvor nach 8000 geretteten Daten überschreiben (zurückholen)
* Dieselbe Wirkung hat die gleichzeitige Betätigung von Control- und Dezimalpunkt-Taste auf der Tastatur-Erweiterung

3. Rechner-Kopplung:

V – Serielle Ausgabe über die CPU-Schnittstelle (mit variabler Übertragungsrate)
W – Serielle Eingabe über die CPU-Schnittstelle (mit variabler Übertragungsrate)

4. Ansteuerung interner Ergänzungs-Karten:

U – Uhrzeit auslesen bzw. einstellen (Echtzeit-Uhr)
P – PROM lesen/programmieren/vergleichen (EPROM-Programmierzusatz)
H – Speicherinhalte hexadezimal ausdrucken (Thermodrucker)
K – ASCII-Zeichenfolge (Klartext) ausdrucken (Thermodrucker)

5. Aktivierung des Einzelschritt-Moduls:

S – Hardware-Einzelschritt-Betrieb mit Anzeige sämtlicher Registerinhalte (im RAM und ROM möglich)
Z – Breakpoint setzen und im Einzelschritt-Betrieb fortfahren (Breakpoint kann nur im RAM-Bereich liegen)
Y – Einzelzyklus-Betrieb mit Anzeige von Daten- und Adreßbus sowie augenblicklichem Prozessor-Zyklus

6. Ansteuerung externer Peripheriegeräte:

O – Kassetten-Output über das schnelle Magnetband-Interface (Universal-E/A-Karte)
I – Kassetten-Input über das schnelle Magnetband-Interface (Universal-E/A-Karte)
N – Bildschirmabzug ausdrucken (großer Drucker; serielle Ansteuerung)
T – Textseite holen (z. B. mit Telefonnummern-Verzeichnis)

7. Aufruf zusätzlicher Programmpakete:

A – Assembler (auf Platz #6 der großen Speicherkarte)
B – BASIC (auf den Plätzen #4 und #5 der großen Speicherkarte)
D – Disketten-Betriebssystem (Ladeprogramm auf Platz #5 der großen Speicherkarte; Software auf Diskette)
E – Text-Editor (auf Platz #7 der großen Speicherkarte)

gibt es die drei Bildschirm-Kommandos zum Löschen, Retten und Zurückholen des Bildschirminhalts; sie sind zum Erstellen von Textzeilen in Anwenderprogrammen vorgesehen, um sich die Quälerei mit den einzelnen ASCII-Codes zu ersparen.
3. Zwei Kommandos sind dafür reserviert, die serielle Datenübertragung von einem Mikrocomputer zum anderen zu veranlassen. Für diese Übertragung sind jeweils der serielle CPU-Eingang des einen mit dem seriellen CPU-Ausgang des anderen Computers zu verbinden.
4. Die systeminternen, eigenständigen Peripherie-Baugruppen Echtzeit-Uhr, EPROM-Programmierzusatz und Thermodrucker lassen sich über eigene Anweisungen aktivieren. Mit der Bildschirmunterstützung entsteht hiermit ein komplettes, kleines Entwicklungssystem.
5. Zusätzlich sind drei Kommandos zur Ansteuerung des Einzelschritt-Moduls vorgesehen, mit dem Programme im Einzelschritt-, Einzelbyte- oder Breakpoint-Betrieb abgearbeitet werden können, was für die Inbetriebnahme und die Schulung eine wesentliche Unterstützung darstellt.
6. Für die externen Peripheriegeräte Matrixdrucker und schnelles Cassetten-Interface sowie zur Verarbeitung von (Bildschirm-)Textseiten sind vier eigene Anweisungen reserviert. Sie dienen vornehmlich Dokumentations- und Archivierungsaufgaben.
7. Und schließlich besteht die Möglichkeit, zu vier weiteren Software-Paketen zu verzweigen (Assembler, Editor, BASIC und Disketten-Betriebssystem), aus denen aber stets der Rücksprung in die Monitor-(Anweisungs-)Ebene erfolgt. Durch diese Organisation gestaltet sich die Handhabung des Systems wesentlich flexibler, als wenn man alle Leistungsmerkmale in eine Programmebene zwängt.
Schauen wir uns die Handhabung der einzelnen Kommandos in der eben angeführten Reihenfolge einmal an; dabei ist zu beachten, daß zwischen dem Buchstaben einer Anweisung und einer nachfolgenden Adresse immer ein Leerschritt einzufügen ist, nicht aber zwischen zwei aufeinanderfolgenden Adressen; die werden (wahlweise) durch ein Komma oder einen Leerschritt voneinander getrennt. Einige Anweisungen lassen auch eine Kurzform ohne Adreßangabe zu; dies ist insbesondere in der Testphase von Vorteil, wo man oft ändern und ausprobieren muß und sich dabei jedesmal die erneute Adreßeingabe sparen kann. Der Rücksprung aus einer angewählten Betriebsart in die Anweisungs-Ebene erfolgt stets durch den (gleichzeitigen) Druck auf „Control" und „C" (dazu also nicht RESET drücken!).

1.1 Maschinensprache:
m xxyy Return

Adresse xxyy aufrufen und den Inhalt darstellen; Inhalt durch Neueingabe modifizieren (nur im RAM-Bereich möglich), wobei das Einschreiben in die angegebene Adresse erst nach dem Weiterschalten zur nächsten Adresse erfolgt; weiterschalten mittels Return *oder* Leerschritt, zurückschalten mittels Minus („–") oder Unterstreichung („_"). Dateneingabe wählen: Pfeil rechts („>"), Adreßeingabe wählen: Pfeil links („<"; **Bild 98**).
Sonderfall: m Return (ohne Adreßspezifikation) ruft Adresse 2800 auf (unterste RAM-Adresse auf der CPU).

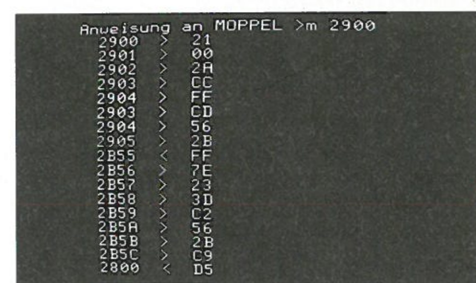


Bild 98: Auch mit Bildschirm und ASCII-Tastatur kann man in Maschinensprache programmieren, und das sogar mit einigem Komfort.



Bild 99: Beim Sprung in ein Anwenderprogramm erfolgt diese Meldung, bei der der blinkende Cursor ausgeschaltet wird.

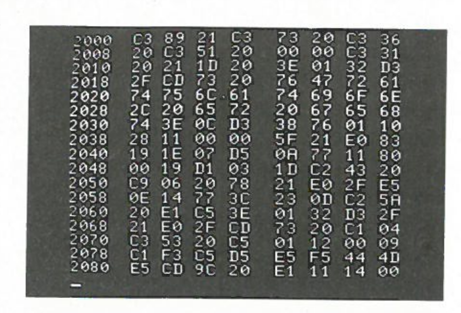


Bild 100: Je nach Bildformat lassen sich 136 bzw. 368 RAM-Zellen auf einmal inspizieren.

```
8180 A5 A5 A5 A5  A5 A5 A5 A5
8188 A5 A5 A5 A5  A5 A5 A5 A5
8190 A5 A5 A5 A5  A5 A5 A5 A5
8198 A5 A5 A5 A5  A5 A5 A5 A5
81A0 A5 A5 A5 A5  A5 A5 A5 A5
81A8 A5 A5 A5 A5  A5 A5 A5 A5
81B0 A5 A5 A5 A5  A5 A5 A5 A5
81B8 A5 A5 A5 A5  A5 A5 A5 A5
81C0 A5 A5 A5 A5  A5 A5 A5 A5
81C8 A5 A5 A5 A5  A5 A5 A5 A5
81D0 A5 A5 A5 A5  A5 A5 A5 A5
81D8 A5 A5 A5 A5  A5 A5 A5 A5
81E0 A5 A5 A5 A5  A5 A5 A5 A5
81E8 A5 A5 A5 A5  A5 A5 A5 A5
81F0 A5 A5 A5 A5  A5 A5 A5 A5
81F8 A5 A5 A5 A5  A5 A5 A5 A5
8200 A5 A5 A5 A5  A5 A5 A5 A5
_
```

Bild 101: Beispielsweise zum Löschen oder für einen Speichertest läßt sich die Funktion „RAM mit Konstante füllen" verwenden.

1.2 Go: g xxyy Return
Programmstart ab Adresse xxyy (**Bild 99**; jede beliebige Adresse ist zulässig). *Sonderfall:* g Return (ohne Adreßspezifikation) bewirkt Programmstart bei Adresse 2800 (unterste RAM-Adresse auf der CPU).

1.3 List: l xxyy Return
Hexadezimale Darstellung des Speicherinhalts ab Adresse xxyy; beim kleinen Bildschirmformat werden 17 Zeilen mit je acht RAM-Zellen dargestellt, beim Großformat sind es 23 Zeilen mit je 16 RAM-Zellen (**Bild 100**). Mit „Return" können Sie sich den nächstfolgenden Speicherblock darstellen lassen, und mit „CTL C" kommen Sie zurück in die Anweisungs-Ebene.

1.4 Fill:
f aaaa,eeee,kk Return
Speicherbereich von aaaa bis eeee mit der Konstanten kk auffüllen (**Bild 101**; nur im RAM-Bereich möglich). Anschließend erfolgt der Sprung in den List-Mode (s. o.) mit Darstellung des Speicherinhalts von der angegebenen Startadresse aaaa an (Funktionskontrolle).

1.5 Copy:
c aaaa,eeee,zzzz Return
Speicherbereich von aaaa bis eeee kopieren in den bei zzzz beginnenden Bereich (der Zielbereich muß im RAM liegen). Anschließend erfolgt der Sprung in den List-Mode (s. o.) mit Darstellung des Speicherinhalts von der angegebenen Zieladresse zzzz an (Funktionskontrolle).

1.6 Exchange:
x aaaa,eeee,zzzz Return
Wie Copy, aber mit Anpassung der absoluten Sprungadressen an den neuen Zielbereich (Verschieben von Programmteilen).

2.1 Bildschirm löschen:
j Return
Video-Speicher mit Leerzeichen (= Blanks, 20 HEX) füllen; automatischer Rücksprung in die Anweisungs-Ebene.

2.2 Retten des Bildschirm-Inhalts: r Return
Inhalt des Video-Speichers ab 3000ff. nach 8000ff. überschreiben; die drei untersten Bildzeilen werden dabei nicht mit übertragen. *Achtung:* Das „r" (oder „R") muß dabei, wie üblich, rechts neben dem Anweisungspfeil in der untersten Zeile stehen; bei vorhandener Tastatur-Erweiterung wird dieses Kommando auch unabhängig von der Cursor-Stellung ausgeführt, wenn Sie (gleichzeitig) „Control" und *den Dezimalpunkt auf der Erweiterung* betätigen.

2.3 Bildschirm-Inhalt zurückholen: q Return
Den zuvor per r-Kommando (s. o.) nach 8000ff. geretteten Bildschirm-Inhalt zurückholen (mit Ausnahme der drei untersten Bildzeilen).

3.1 Serielle Ausgabe:
v aaaa,eeee,bbbb Return
Speicherbereich von aaaa bis eeee mit der Baudrate bbbb über die serielle CPU-Schnittstelle ausgeben (**Bild 102**). Die Übertragung beginnt erst dann, wenn die Bereitmeldung der angeschlossenen Empfangsstelle erfolgt (im w-Mode; s. u.). Während der Übertragung ist der Cursor ausgeschaltet, und er blinkt erst wieder nach beendeter Datenausgabe. Liegt empfangsseitig eine Fehlermeldung vor, kann eine erneute Übertragung einfach durch Druck auf „Return" ausgelöst werden. In den Anweisungs-Mode kommen Sie mit „CTL C" zurück. Mögliche Übertragungsraten: 110, 300, 600, 1200, 2400, 4800 und 9600 Baud.

3.2 Serielle Eingabe:
w aaaa,eeee,bbbb Return
Über die serielle CPU-Schnittstelle in den Speicherbereich von aaaa bis eeee mit der Baudrate bbbb einlesen (**Bild 103**); vor dem Starten muß die angeschlossene Sendestelle im v-Mode sein (s. o.). Übertragungsfehler werden vom Programm gemeldet, während bei fehlerfreier Übertragung der Sprung in den List-Mode erfolgt (s. o.). Mögliche Übertragungsraten: 110, 300, 600, 1200, 2400, 4800 und 9600 Baud.

4.1.1 Lesen der Echtzeit-Uhr: u Return
Darstellen von Datum und Uhrzeit auf dem Bildschirm (**Bild 104**). *Achtung!* Zum Lesen darf außer dem „u" (oder „U") kein weiteres Zeichen eingegeben werden, weil das Programm dies als Anweisung zum Stellen der Uhr interpretieren würde.

4.1.2 Stellen der Echtzeit-Uhr: u ssmm,wkk,mmjj Return
Eingabe der Stunden/Minuten ssmm, des Wochentags w (1...7, wobei die Eins dem Montag entspricht) und Kalendertags sowie des Monats und Jahrs (die Sekunden werden automatisch auf Null gesetzt). Beispiel: für Sonntag (= 7), den 6. 3. 83, 14:28 müssen die Daten gemäß Bild 104 (unten) eingegeben werden (mit Return abschließen). Zur Kontrolle erfolgt nach dem Stellen der Uhr sofort das Auslesen und die Darstellung auf dem Bildschirm (ohne Hochzählen der Zeit).

4.2.1 PROM lesen:
p aaaa,eeee,zzzz Return
Typ eingeben: **16** oder **32** Return (Umschalter auch auf 16/32)

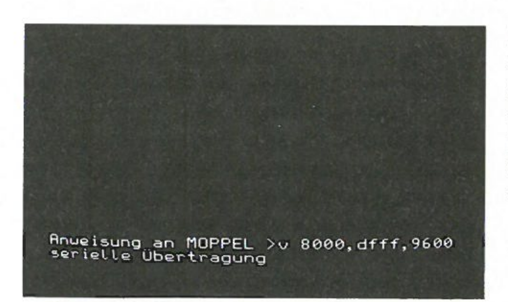


Bild 102: Über die serielle CPU-Schnittstelle können sich zwei Computer „unterhalten", d. h. Daten miteinander austauschen.

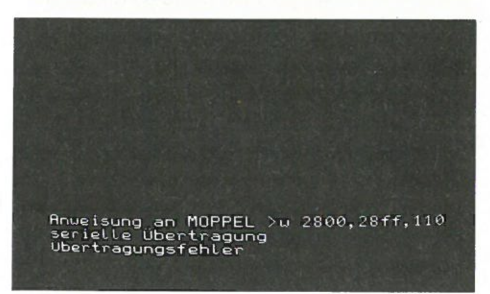


Bild 103: Beim Einlesen erfolgt automatisch eine Prüfung der empfangenen Daten mit Fehlermeldung bei negativem Prüfergebnis.

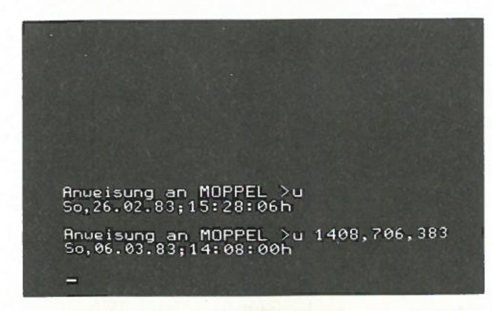


Bild 104: Das Auslesen (oben) und Stellen (unten) der Echtzeit-Uhr erfolgt beide Male mit dem u-Kommando.

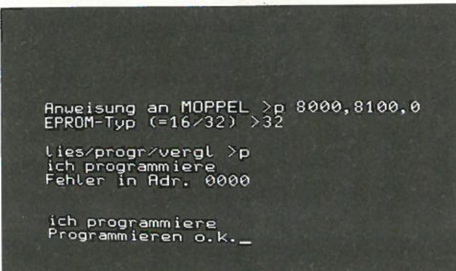


Bild 105: Beim Auslesen, Programmieren und Vergleichen von EPROMs spielt sich ein regelrechter Dialog ab.

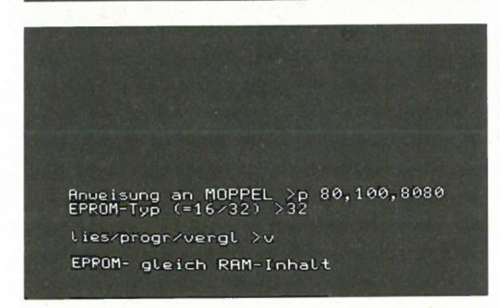
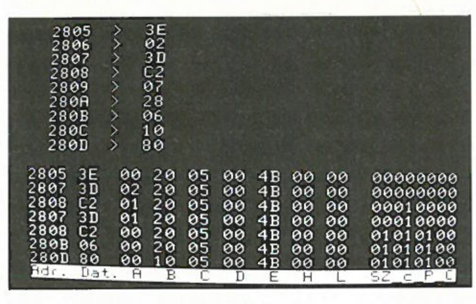


Bild 106: Das Überprüfen vorhandener, programmierter EPROMs gestaltet sich durch den Vergleichs-Mode sehr unkompliziert.



Adr.	Dat.	A	B	C	D	E	H	L	SZ c P C
2805	3E	00	20	05	00	4B	00	00	00000000
2807	3D	02	20	05	00	4B	00	00	00000000
2808	C2	01	20	05	00	4B	00	00	00010000
2807	3D	01	20	05	00	4B	00	00	00010000
2808	C2	00	20	05	00	4B	00	00	01010100
280B	06	00	20	05	00	4B	00	00	01010100
280D	80	00	10	05	00	4B	00	00	01010100

Bild 107: Mit dem Einzelschritt-Modul kann man Programme schrittweise verfolgen und dabei sämtliche Register-Inhalte inspizieren.

Lesen wählen: **l** Return
Aus dem EPROM des genannten Typs (2716/2732) den Adreßbereich aaaa bis eeee ins RAM ab Adresse zzzz überschreiben. Anschließend erfolgt der Sprung in den List Mode (s. o.), um das Einlesen kontrollieren zu können. In den Anweisungs-Mode kommen Sie mit „CTL C" zurück.

4.2.2 PROM programmieren:
p aaaa,eeee,zzzz
Return
Typ eingeben: **16** oder **32** Return (Umschalter auch auf 16/32)
Programmieren wählen: **p** Return
Den Speicherbereich aaaa bis eeee ins EPROM des genannten Typs (2716/2732) übertragen, beginnend bei PROM-Adresse zzzz (**Bild 105**). Nach dem Programmieren eines Bytes erfolgt unmittelbar die Kontrolle, ob das Einschreiben fehlerfrei funktioniert hat. Liegt ein Fehler vor, wird der Programmiervorgang abgebrochen und die zugehörige Adresse gemeldet (Bild 105, Mitte). Nach Einsetzen eines neuen EPROMs können Sie den Programmiervorgang mit denselben Adressen erneut starten, indem Sie einfach „Return" drücken. Das Programm meldet auch die fehlerfreie Programmierung (Bild 105, unten). In den Anweisungs-Mode kommen Sie mit „CTL C" zurück.

4.2.3 PROM mit RAM vergleichen:
p aaaa,eeee,zzzz
Return
Typ eingeben: **16** oder **32** Return (Umschalter auch auf 16/32)
Vergleichen wählen: **v** Return
Den Adreßbereich von aaaa bis eeee des genannten EPROM-Typs (2716/2732) mit dem Speicherbereich ab zzzz vergleichen. Gleichheit oder die erste abweichende Adresse werden vom Programm gemeldet (**Bild 106**). In den Anweisungs-Mode kommen Sie mit „CTL C" zurück.

4.3 Hexadezimaler Speicherabzug: h aaaa,eeee
Return
Den Speicherbereich von aaaa bis eeee hexadezimal auf dem Thermodrucker ausdrucken. Eingegebene Anfangs- und Endadresse werden dabei auf glatte Werte nach unten bzw. nach oben gerundet. Der Rücksprung in den Anweisungs-Mode erfolgt automatisch.

4.4 Klartext ausdrucken:
k aaaa,nn Return
Den Speicherinhalt (ASCII-Zeichen) ab aaaa auf dem Thermodrucker ausdrucken (nn Druckzeilen mit je 20 Zeichen). Der Rücksprung in den Anweisungs-Mode erfolgt automatisch.

5.1 Single Step-(Einzelschritt-)Betrieb:
s xxyy Return
Das Programm ab Adresse xxyy befehlsweise abarbeiten und dabei alle Registerinhalte und Zustandssignale (FLAGs) anzeigen (**Bild 107**). Nur zusammen mit dem Einzelschritt-Modul durchführbar, allerdings ohne Einschränkung hinsichtlich der Unterprogramm-Verschachtelung oder der Anordnung im RAM bzw. EPROM.

5.2 Breakpoint-(Haltepunkt-)Betrieb:
z aaaa,eeee Return
Das Programm von Adresse aaaa bis eeee im Echtzeit-Betrieb durchlaufen, danach weiter im Einzelschritt-Mode (s. o.). Nur zusammen mit dem Einzelschritt-Modul und Programmen im RAM durchführbar.

5.3 Einzel-Zyklus-Betrieb:
y aaaa Return
Das Programm ab Adresse aaaa zyklusweise abarbeiten und dabei den Zustand auf dem Daten- und Adreßbus sowie den Typ des jeweiligen Prozessor-Zyklus anzeigen (**Bild 108**). Nur zusammen mit dem Einzelschritt-Modul durchführbar, allerdings ohne Einschränkung hinsichtlich der Unterprogramm-Verschachtelung oder der Anordnung im RAM bzw. EPROM.

6.1 Ausgabe über Magnetband-Interface:
o aaaa,eeee Return
Den Speicherbereich von aaaa bis eeee über das schnelle

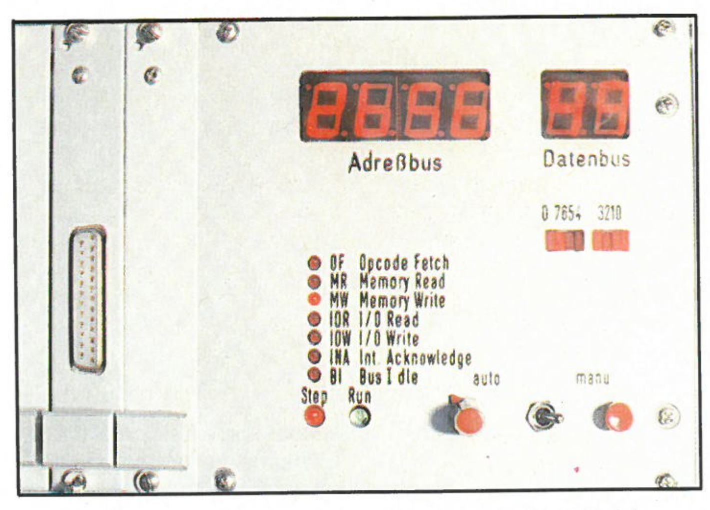


Bild 108: Noch eine Ebene tiefer dringt der Einzelzyklus-Betrieb vor: Hier wird jeder einzelne Prozessor-Zyklus analysiert.

Cassetten-Interface (auf dem seriellen Interface) ausgeben. Der Anschluß des Bandgerätes erfolgt an der Tonbandbuchse der Bus-Platine. Bei fehlerfreier Übertragung (während der der Cursor ausgeschaltet ist) springt das Programm automatisch zurück in die Anweisungs-Ebene, andernfalls erzeugt es eine Fehlermeldung.

6.2 Eingabe über Magnetband-Interface:
i aaaa,eeee Return
Vom schnellen Cassetten-Interface (auf dem seriellen Interface) Daten in den Speicherbereich von aaaa bis eeee einlesen. Der Anschluß des Bandgerätes erfolgt an der Tonbandbuchse der Bus-Platine. Bei fehlerfreier Übertragung springt das Programm in den List-Mode (s. o.), andernfalls erzeugt es eine Fehlermeldung (vgl. Bild 103).

6.3 Bildschirminhalt ausdrucken: n Return
Ansteuerung eines externen Druckers, der an der 25poligen Bus-Steckbuchse anzuschließen ist und einen Bildschirmabzug ausdruckt. Der Rücksprung in die Anweisungs-Ebene erfolgt automatisch.

6.4 Textseite holen:
t n Return
Textseite Nr. n (= 0...F) aus dem Speicher holen und auf dem Bildschirm darstellen. Der Rücksprung in die Anweisungs-Ebene erfolgt durch „CTL C".

7.1 Assembler aufrufen:
a Return
Programmentwicklung mit einem mnemonischen 8085-Assembler in EPROM (große Speicherkarte erforderlich). Der Rücksprung in die Anweisungs-Ebene erfolgt durch „CTL C".

7.2 BASIC aufrufen: b Return
Sprung in das 8-K-MOPPEL-BASIC im EPROM (Microsoft-kompatibel, durch einige Zusatzfunktionen erweitert; große Speicherkarte erforderlich). Der Rücksprung in die Anweisungs-Ebene erfolgt durch das BASIC-Direktkommando „bye".

7.3 Disketten-Betriebssystem aufrufen: d Return
Einlesen des Betriebsprogramms von Diskette (große Speicherkarte und Floppy-Controller erforderlich). Der Rücksprung in die Anweisungs-Ebene erfolgt durch „CTL C".

7.4 Editor aufrufen: e Return
Text-Editor zum Erstellen und Modifizieren von Texten (große Speicherkarte erforderlich). Der Rücksprung in die Anweisungs-Ebene erfolgt durch „CTL C".

Natürlich werden wir die einzelnen Betriebsarten noch ausführlich am Beispiel erläutern, da sie ja z. T. ganz umfassende Funktionen beinhalten, wie z. B. die Programmierung in BASIC.

Die Basis für BASIC

Wenn Sie sich, ganz gleich aus welchem Grund, mit der höheren Programmiersprache BASIC befassen wollen, dann müssen Sie dazu zwei Dinge kennen: Erstens einmal die Elemente dieser Sprache (quasi die Vokabeln) und zweitens die Randinformationen darüber, wie man diese Elemente miteinander verknüpft. Sie brauchen jetzt keine Angst zu haben, daß wir Sie hier, ähnlich wie die Schule, mit Vokabel-Büffeln überfallen; das gute an BASIC ist, daß man es (fast) von allein begreift und erlernt, wenn man ein paar Grundregeln beherzigt. Dazu ist es vier, fünf Mal erforderlich, daß Sie sich bestimmte, bislang ungewohnte Zusammenhänge mit dem Verstand klar machen, daß Sie diese Zeilen also nicht blind, sondern wachen Sinnes nachvollziehen. Wir versprechen Ihnen dafür, daß Sie in ganz kurzer Zeit auch komplexe Probleme virtuos in ein Programm umsetzen können. Wenn man dabei richtig vorgeht, macht das Ganze einen Heidenspaß; unser Ziel ist es, Ihnen diesen richtigen Weg zu vermitteln. Zu Ihrer Information: „BASIC" ist ein Kunstwort, das vom amerikanischen *Beginner's All Purpose Symbolic Instruction Code* abgeleitet ist und soviel bedeutet wie „vielseitige, symbolische Programmiersprache für Anfänger", was aber keine Abqualifikation bedeutet, denn auch Profis finden mit BASIC ein weites Auslaufgebiet (**Bild 109**). Mit diesem Computer-Chinesisch kann ein Neuling naturgemäß nicht viel anfangen, darum noch einmal etwas deutlicher: Mit Hilfe dieser Programmiersprache reden Sie mit Ihrem Computer auf einer höheren, d. h. der menschlichen Sprachform näheren Ebene, z. B. so: **Wenn** die Zahl **x < 15** ist, **gehe zu** Befehl Nr. **4711;** andernfalls mache beim nächstfolgenden Befehl weiter. Beim Programmieren genügt es, nur die fettgedruckten Elemente einzugeben; den Rest denken Sie (und der Computer) sich dazu. Es interessiert Sie also nicht mehr die CPU-interne Registerstruktur, sondern Sie taufen Ihre Zahlen mit Symbolen, denen der Computer entsprechende Zahlenwerte zuordnet, Vorzeichen, Nachkommastellen und Exponent eingeschlossen. Damit wird Ihr Mikrocomputer schlagartig zu einer richtigen *Rechenmaschine*, die jedem handelsüblichen Taschenrechner überlegen ist. Das zweite hervorstechende Merkmal von BASIC ist die Möglichkeit der flexiblen *Behandlung von Texten,* die bis zur Buchstaben-Ebene herunterreicht, d. h. es gibt Befehle, mit denen man einzelne Buchstaben in einer Zeichenkette abfragen oder modifizieren kann. Diese beiden wesentlichen Eigenschaften des Rechnens und der Textbearbeitung machen einen Computer eigentlich erst zu einem richtigen Computer, der Klartext-Anweisungen entgegennimmt und sich bei Rückmeldungen ebenso verständlich ausdrückt. Sie sollten sich aber davor hüten, BASIC als das Nonplusultra anzusehen; es ist zweifellos eine liebenswerte Beschäftigungsmöglichkeit

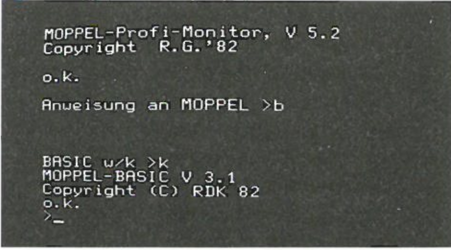


Bild 109: Gleichzeitig mit der Bereitmeldung erscheint die Version des BASIC-Interpreters (hier Version 3.1).

```
Anweisung an MOPPEL >b

BASIC u/k >k
MOPPEL-BASIC V 3.1
Copyright (C) RDK 82
o.k.
>new

o.k.
>100 call hex("446")
>list

100 CALL HEX("446")
o.k.
>
```

Bild 110: Einige BASIC-Befehle bewegen sich auf Maschinen-Ebene, z. B. der Aufruf eines Maschinen-Unterprogramms mit Angabe der hexadezimalen Startadresse

mit dem Computer, die einem Neuling aber sehr schnell den Programmierverstand verderben kann, wenn er nicht diszipliniert zu Werke geht.
Es gibt nun nicht *das* BASIC schlechthin, sondern einen BASIC-Grundbefehlssatz, der von den verschiedenen Computer-Herstellern unterschiedlich erweitert und an den jeweiligen Maschinen-Typ angepaßt ist; man spricht in diesem Zusammenhang von verschiedenen „BASIC-Dialekten", und so hat auch das für den MOPPEL geschriebene BASIC Feinheiten, die es von anderen „BASICs" unterscheidet, die aber für unsere Belange von ganz entscheidender Bedeutung sind. Für alle Fachleute und Schlagwort-Sammler: Das MOPPEL-BASIC entspricht weitgehend der 8-K-Microsoft-Version, ergänzt durch maschinennahe Befehle, die auch noch in der höheren Programmiersprache ständig den Bezug zur Computer-Hardware herstellen. Und das ist, wie Sie noch sehen werden, ein nicht zu unterschätzender Vorteil (**Bild 110**)! Kompatibilität (d. h. Übertragbarkeit) zu anderen BASIC-Computern besteht insoweit, als man die Programme durch Neueintippen übernehmen kann und nichtvorhandene Befehle nachbildet (das ist in den meisten Fällen ziemlich problemlos möglich). Wegen der unterschiedlichen Hardware-Konfiguration ist es bei Maschinen unterschiedlichen Typs in der Regel nicht möglich, Programme direkt zu überspielen, beispielsweise über die Tonbandbuchsen. Gerade für derartige Programmanpassungen ist es wiederum erforderlich, die eigene Hardware genau zu kennen und Befehle zur Verfügung zu haben, um gezielt darauf zuzugreifen (vgl. ELO-Laborbrief Nr. 2).

Beim Abschied heißt es „tschüß"

Der BASIC-Interpreter für den MOPPEL ist in zwei 4-K-EPROMs untergebracht, die auf den Plätzen #4 und #5 der großen Speicherkarte residieren (**Bild 111**). Damit degradieren Sie Ihren MOPPEL aber noch lange nicht zum reinen BASIC-Computer, sondern Sie erweitern seinen Sprachschatz nur um diese Ebene. Der BASIC-Aufruf erfolgt über „b" und Return, woraufhin Sie die Frage gestellt bekommen: „BASIC w/k >", d. h. ob Sie einen BASIC-Warm- oder -Kaltstart wünschen. „Kalt" müssen Sie nach jedem Einschalten der Maschine starten („k", Return), damit sich BASIC erst einmal selbst ordnen kann, nachdem es den freien Speicherraum (ab $8000) ermittelt hat. Nach dem Verlassen von BASIC (dem Rücksprung in die Monitor-Anweisungs-Ebene per Direkt-Kommando „bye") können Sie BASIC aufrufen, *ohne den Speicher zu löschen und damit vorhandene Programme zu zerstören*, indem Sie „warm" starten („w", Return). Ein laufendes BASIC-Programm *stoppen* Sie, indem Sie eine beliebige Taste drükken; nach dem Loslassen geht die Programmausführung kommentarlos an der unterbrochenen Stelle weiter. Ein solches Programm *brechen Sie ab*, indem Sie gleichzeitig Control und C betätigen; Sie gelangen damit in die BASIC-Kommando-Ebene zurück, in der Sie z. B. Programme eingeben oder ändern können (Achtung! Nicht alle Befehle lassen sich auf diese Weise unterbrechen, z. B. INPUT nicht). Zurück in die MOPPEL-Monitor-Ebene kommen Sie, wie erwähnt, mittels „bye", was etwa dem „Tschüß" entspricht.
Und da wir gerade beim Englischen sind: BASIC kommt aus Amerika, und dementsprechend setzt es sich aus englischen Sprachelementen zusammen; denken Sie jetzt nicht mit Grauen an Ihre Englisch-Fünf in der Schule, sondern sagen Sie sich, daß die paar englischen Brocken, die in BASIC vorkommen, jeder halbgare Neandertaler erlernen könnte, und daß ein BASIC mit deutschen Befehlen zwar jederzeit möglich, aber wenig sinnvoll wäre, weil es von der internationalen Norm abweicht. Aus diesem Grund haben wir es beim Original-Befehlssatz belassen, die Klartext-Rückmeldungen aber ins Deutsche übertragen, um allen Sprachgeschädigten wenigstens ein bißchen entgegenzukommen (**Bild 112**).
Sämtliche Befehle können Sie in Kleinbuchstaben eingeben, sie werden beim internen Übersetzungsvorgang automatisch in Großschreibung umgesetzt. Hiervon gibt es zwei Ausnahmen, die Sie unbedingt beachten müssen: Das „E" (für „Exponent", s. u.) und die Zeichen „A...F" bei hexadezimalen Zahlen *müssen* in jedem Fall als Großbuchstaben eingegeben werden, weil BASIC sonst einen (Syntax-) Fehler anmeckert.

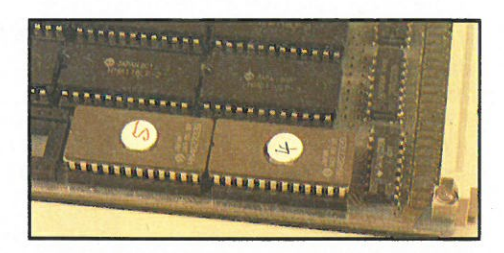


Bild 111: Die beiden 4-K-EPROMs mit dem BASIC-Interpreter müssen auf den Plätzen #4 und #5 auf der großen Speicherkarte eingesetzt werden.

```
MOPPEL-BASIC V 3.1
Copyright (C) RDK 82
o.k.
>new

o.k.
>100 call hex("446")
>list

100 CALL HEX("446")
o.k.
>100 input a
>run
? 2,5,7
das war etwas zuviel

o.k.
>_
```

Bild 112: Bei den Klartext-Hinweisen meldet sich das MOPPEL-BASIC in deutsch.

Hausnummern für die Befehle

Es ist hier mehrfach dezent von „Ebenen" der Programmierung geredet (bzw. geschrieben) worden; BASIC unterscheidet zwei Ebenen, nämlich die Programm-Ebene, in der das eigentliche Programm abläuft sowie die Kommando-Ebene, in der man Programme eingibt, inspiziert oder ändert und sonstige Verwaltungsaufgaben vornimmt. Will man eine Anweisung in ein Programm einbauen, muß man ihr eine *Zeilennummer* voranstellen; die Programmzeilen werden dann in aufsteigender Reihenfolge abgearbeitet, wobei die Numerierung zweckmäßigerweise nicht lükkenlos, sondern z. B. in Zehner-Intervallen erfolgt, um später noch Einfügungen dazwischenquetschen zu können (so einfach geht das in Maschinensprache nicht!). Die höchste zulässige Zeilennummer ist übrigens 65529, was aber noch lange nicht heißt, daß auch so viele Zeilen im Speicher Platz haben; man kann aber bereits bei der Vergabe der Zeilennummern eine Gliederung hineinbringen, bei der z. B. die Hunderter, Tausender und Zehntausender verschiedene Unterprogramm-Verschachtelungen kennzeichnen (das ist vielfach schon die halbe Dokumentation!).
Es gibt viele erfolgversprechende und etliche unmögliche Wege, sich in BASIC einzuarbeiten. Wir schlagen Ihnen dazu einen Weg vor, bei dem Sie durch ständige Erfolgserlebnisse mit Spaß am Ball bleiben, stets die Übersicht behalten und kennenlernen, wie man einer Problemlösung fundiert zu Leibe rückt. Sie bekommen deshalb keine trockene Einführung vorgesetzt, sondern sofort ein „echtes" Anwendungsbeispiel, um das herum sich die jeweiligen Befehle ranken. Vollziehen Sie jeden einzelnen Schritt nach,

aber jeden weiteren immer erst nach Verständnis des vorhergehenden; Sie werden erstaunt sein, wie schnell Sie dabei firm in dieser Sprache werden, die Ihnen bald in Fleisch und Blut übergeht!
Die Standard-Ausgabe-Anweisung in BASIC lautet „PRINT" (bzw. „print"), was zwar wörtlich mit „drucke" zu übersetzen ist, tatsächlich aber zur Darstellung auf dem Bildschirm führt. Wenn Sie eingeben „PRINT 3", gefolgt von Return, schreibt Ihr Mops eine „3" auf den Bildschirm. Da die PRINT-Anweisung andauernd vorkommt, hat man dafür eine Abkürzung eingeführt, und zwar das Fragezeichen (warum es ausgerechnet das Fragezeichen geworden ist, bleibt ein wohlgehütetes Geheimnis); die Eingabe „? 3", abgeschlossen mit Return führt demnach ebenso zur Ausgabe der „3" auf dem Bildschirm. Da das noch wenig aufregend ist, versuchen Sie es mal mit einfachen Rechenaufgaben. Das Multiplikationszeichen in BASIC heißt „*", das Divisionszeichen „/". Bei „? 47*11", Return erfahren Sie sofort das Ergebnis „517" auf dem Bildschirm, und „? 47/11", Return führt zur Anzeige „4.2727". Dabei erkennen Sie sofort eine wichtige Kleinigkeit: Das gewohnte Komma in Dezimalzahlen (4,2727) wird in BASIC zum Pünktchen; *das Komma ist als Trennzeichen zwischen zwei Zahlen reserviert,* so daß der Computer die gutgemeinte Eingabe von 2,5 (für Zweieinhalb) mißversteht und daraus eine Zwei macht, gefolgt von einer Fünf.

Der Mops fragt seinen Meister

Das Gegenstück zu PRINT (Ausgabe) ist „INPUT" (bzw. „input" = Eingabe); dieser Befehl verlangt vom Anwender eine Eingabe über die Tastatur, die, wie allgemein üblich, mittels Return abzuschließen ist. Angenommen, Sie wollen (allgemein ausgedrückt) zwei Zahlen a und b miteinander multiplizieren und diese beiden Zahlen über die Tastatur vorgeben; dann schreiben Sie das folgende dreizeilige BASIC-Programm

```
10 input a,b
20 c = a*b
30 print c (abgekürzt: 30 ?c)
```

Durch das Voranstellen der Zeilennummern wird die Befehlsfolge zum Programm, während wir uns im Beispiel oben (bei „? 47/11" z. B.) noch in der Kommando-Ebene („Direkt-Modus") bewegt haben, bei der nur eine einzige Anweisung zur Zeit ausgeführt werden kann. Ein BASIC-Programm startet man per „RUN" (bzw. „run"), gefolgt von Return. Im Beispiel hier erwartet das Programm von Ihnen die Eingabe der Zahlen a und b, was Sie am Fragezeichen auf dem Bildschirm erkennen (Zeile 10). Sie müssen nun für a und b jeweils eine Zahl eingeben, beide durch ein Komma trennen und mit Return abschließen, z. B.:

? 64,59 Return

Das Programm setzt dann für „a" den ersten Wert 64 und für „b" den zweiten Wert, in diesem Fall 59, ein, bildet das Produkt, das es fortan unter der (zufällig gewählten) Bezeichnung „c" führt (alles Programmzeile 20) und gibt dann das Ergebnis auf dem Bildschirm bekannt (im Beispiel sind das 3776). Zur Endlosschleife wird dieses Programmchen, wenn Sie eine weitere Zeile anfügen: 40 goto 10; dann können Sie sich einen Abend vor Ihren Computer setzen und das große Einmaleins kontrollieren. Natürlich können Sie für a oder b auch Dezimalzahlen eingeben, wenn Sie nur darauf achten, statt des bei uns üblichen Dezimal*kommas* ein Pünktchen einzugeben, denn das Komma trennt ja die beiden Zahlen voneinander (**Bild 113**).
Bis hierhin haben Sie möglicherweise gelangweilt mitgelesen, ohne sich Gedanken darüber zu machen, welch fundamentaler Sachverhalt hinter diesen banalen Zusammenhängen steht. Unser Mini-Programm hat nämlich keine absoluten Zahlen, sondern stattdessen die *Variablen* a, b und c benutzt, die, je nach Eingabe, ganz unterschiedliche Werte annehmen können. In diesem Fall der Zahlenverarbeitung handelt es sich um *numerische Variablen*, und daneben kennt BASIC noch sogenannte *String-Variablen* (Zeichenketten-Variable), hinter denen sich eine beliebige Folge von ASCII-Zeichen verbirgt; eine String-Variable wird durch ein nachgestelltes Dollar-Zeichen kenntlich gemacht, z. B. „a$".

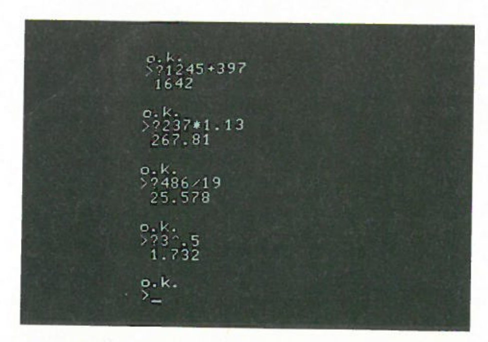


Bild 113: In BASIC wird Ihr Computer unmittelbar zur Rechenmaschine.

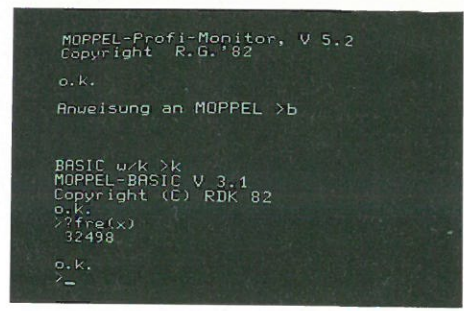


Bild 114: Die in manchen Befehlen auftauchende Dummy-Variable kann jeden beliebigen Wert annehmen, weil dasProgramm dafür keinen Speicherplatz reserviert.

Auf ein Neues

Löschen Sie das Programm von eben, indem Sie eingeben „NEW" (bzw. „new"), gefolgt von Return; das weist den Computer an, den Speicher komplett zu löschen und sich für eine Neueingabe bereitzuhalten. Ab sofort wollen wir darauf verzichten, das obligate „Return" jedesmal zu erwähnen sowie Groß- und Kleinschreibung nebeneinander zu nennen; beides wissen Sie mittlerweile und ergänzen es in Gedanken (und auf der Tastatur). Wenn Sie übrigens nur eine einzelne Zeile löschen wollen, genügt es, die betreffende Zeilennummer einzugeben, gefolgt von Return; in diesem Fall steht in der Zeile nichts mehr, und der Computer vergißt sie deshalb.
Das folgende Programm erwartet von Ihnen wiederum eine Eingabe, diesmal aber die eines Textes; der darf zwar Ziffern enthalten, aber rechnen kann das Programm damit nicht, weil es sich bei „a$" um eine Zeichenfolge handelt und nicht um eine Zahl:

```
10 input a$
20 print a$ (abgekürzt: ?a$)
30 goto 10
```

Nach „RUN" können Sie das schmachtende Fragezeichen des Computers mit jeder beliebigen Eingabe beantworten; in Zeile 20 gibt Ihnen die brave Maschine dann den ganzen Salat wieder aus.
Um in einer Zeile auf dem Bildschirm zwei (oder mehr) *Ausgaben nebeneinander* anzuordnen, führt man dazwischen ein Semikolon ein; die Anweisung „? 2;5" führt demnach zur Ausgabe „2 5". Zwischen beiden Zahlen stehen übrigens zwei Leerzeichen (Blanks); das erste dient als Trennzeichen, und bei positiven Zahlen steht statt eines „+" stets ein Leerzeichen vor der Zahl (bei der Länge von Strings zu beachten!). Außerdem müssen Sie wissen, daß für sämtliche Variablen so lange der Wert Null eingesetzt wird, wie das Programm nichts anderes festlegt. Wenn Sie also nur eingeben „? a", dann registriert das Programm zwar die Variable a, der damit aber noch kein bestimmter Wert zugewiesen wird; dem-

zufolge erscheint bei der PRINT-Anweisung („?“) auf dem Schirm eine „0“. Variablennamen können zweistellig sein, Buchstaben und Ziffern enthalten, sie müssen aber immer mit einem Buchstaben beginnen.

Immer variabel bleiben

Nachdem wir bisher so viel von Variablen geredet haben, fehlt Ihnen noch die Erklärung für die ominöse „Dummy-Variable“ (sprich „dammie“; soviel wie „Pappkamerad“ oder „Muster ohne Wert“), die in der Befehlsübersicht hin und wieder auftaucht. Darunter ist zu verstehen, daß von der Struktur her eine Variable im Befehl auftauchen muß, daß das Programm dafür aber keinen Platz reserviert, diese Variable also nicht weiter in seiner Liste führt (**Bild 114**).
Um in einer Programmzeile zwei (oder mehr) Befehle unterzubringen, führt man dazwischen einen Doppelpunkt ein; wenn Sie also eingeben „a = 2:?a“, erfolgt vor der Ausgabe („?“) erst die Wertzuweisung „a = 2“. In diesem Fall erscheint auf dem Schirm folglich eine „2“. Natürlich können Sie alle bisher genannten Feinheiten sowohl in ein Programm einbauen als auch im Direkt-Modus aufrufen (ohne vorangestellte Zeilennummer). Zur Veranschaulichung stellt die **Tabelle 8** noch einmal die verschiedenen Reaktionen bei der PRINT-Anweisung zusammen, weil diese beim Programmieren von ganz entscheidender Bedeutung sind. Eine Feinheit ist übrigens noch zu ergänzen: Wenn Sie (vielleicht nur aus Versehen) statt „?2;5“ eingeben „?2,5“, dann tauchen auf dem Bildschirm zwar auch die „2“ und die „5“ auf, aber sie sind so weit voneinander entfernt, als hätten sie was gegeneinander. Das kommt von dem Komma zwischen beiden, das als Tabulator wirkt und die darauffolgende Ausgabe um eine Tabulator-Position (14 Stellen) nach rechts rückt.

Tabelle 8. Unterschiedliche Reaktionen bei der PRINT-Anweisung

Befehl	Ausgabe	Kommentar
PRINT 2	2	Die Zwei wird als Ganzzahl erkannt und als solche ausgegeben
PRINT a	0	Der Variablen a ist noch kein Wert zugewiesen worden; das Programm setzt deshalb Null dafür ein
a=2:PRINT a	2	Vor der Ausgabe-Anweisung wird a der Wert 2 zugewiesen, der dann auch erscheint zwei Befehle in einer Zeile)
PRINT "a"	a	Die in Anführungszeichen stehenden Zeichen erscheinen bei der Ausgabe unverändert
PRINT a$		Der String-Variablen a$ ist noch keine Zeichenkette zugewiesen worden; das Programm setzt in diesem Fall für a$ ein Leerzeichen ein (Blank)
a$=ELO:PRINT a$	ELO	Vor der Ausgabe-Anweisung wird a$ die Zeichenkette "ELO" zugeordnet, die dann auch erscheint (zwei Befehle in einer Zeile)
PRINT 2.5	2.5	Format für die Dezimalzahl 2,5
PRINT 2;5	2 5	Das Semikolon weist das Programm an, mehrere Ausgaben hintereinander in eine Zeile zu schreiben; zwischen beiden Ausgaben wird zur Trennung ein Leerzeichen eingefügt, und bei positiven Zahlen tritt für das fehlende "+" ein weiteres Leerzeichen auf.
PRINT 2,5	2 5	Die zweite Ausgabe steht um eine Tabulatorposition (14 Stellen) nach rechts verschoben (hier nicht maßstäblich wiedergegeben)

Die Verarbeitung von Zahlen erfolgt (einschließlich Dezimalpunkt) sechsstellig plus Vorzeichen plus zweistelliger, vorzeichenbehafteter Exponent. Hunderttausend behandelt das Programm folglich noch als „100 000“, während die Million schon etwas ungewohnt aussieht: „1E+06“; das ist zu lesen als Ein mal Zehn hoch 6“, und in diesem Stil geht es weiter bis maximal ±9.9999E±37, größere (bzw. kleinere) Zahlen verkraftet BASIC nicht mehr. Übrigens weigert sich BASIC beharrlich, nichtsignifikante Nullen auszudrucken; wenn beispielsweise die Zahl 2.5000 auftaucht, wird daraus schnörkellos „2.5“. Beim Anlegen von Tabellen stört das unter Umständen erheblich (wegen der unterschiedlichen Anzahl von Nachkommastellen), aber es gibt Tricks, die hier Abhilfe schaffen und beliebig viele Nullen anhängen. Ja, wer es unbedingt will, kann seine Ergebnisse auch mit Dezimal*komma* darstellen (allerdings sind das dann keine Zahlen mehr, aber das brauchen Sie dem Programm ja nicht zu sagen!).
Zur Verknüpfung von Zahlen und für Abfragen stehen folgende Rechenzeichen („Operatoren“) zur Verfügung: +,−,* und / für Addition, Subtraktion, Multiplikation und Division, sowie ∧ für die Exponentiation. Die Vergleichszeichen <,> und = setzen Sie so ein, wie Sie es sprechen: „Falls x kleiner oder gleich 24 ist, gehe zur Programmzeile 4711“ wird zu „If x<=24 goto 4711“. Nach demselben Schema läuft die Formulierung für „ungleich“ ab: Sie setzen einfach die beiden Operatoren „>“ und „<“ hintereinander: „If x<>0 goto 4711“; gehe zur Zeile 4711, wenn x ungleich Null ist.
Mit dem Gleichheitszeichen ist das übrigens so eine Sache; es hat nämlich beim Programmieren genau die entgegengesetzte Aufgabe wie sonst und ist als Wertzuweisungs-Operator zu verstehen: „x=x+1“ treibt einem Mathematiklehrer das kalte Grauen in die Stirn, weil das alles andere ist als gleich. Beim Programmieren ist mit dieser Formulierung gemeint, daß x ab sofort ein neuer Wert zugewiesen wird, und zwar einer, der um Eins größer ist als der alte. Es kann durchaus vorkommen, daß einer Variablen innerhalb eines Pro-

gramms unterschiedliche Werte zugewiesen werden. In diesem Fall gewinnt dann immer die letzte, d. h. es wird mit demjenigen Wert weitergearbeitet, den die Variable zuletzt zugeordnet bekommen hat.

Machen Sie sich mit diesem Grundwissen nun daran, unseren am praktischen Beispiel aufgehängten BASIC-Grundkurs („BASIC für Beginner" ab ELO 6/'83) nachzuvollziehen. Probieren Sie beim Abtippen der Programmschritte ruhig mal etwas anderes aus als beschrieben, damit Sie sicher werden im Umgang mit den Befehlen. Tun Sie sich aber bitte in jedem Augenblick den Gefallen, ein klares Konzept zu entwerfen und zielstrebig zu verfolgen, so wie wir es Ihnen vormachen. BASIC bietet sich nämlich an, schnell das Hudeln anzufangen und wüst hin- und herspringende Programme zu kreieren. Weil das der sicherste Weg ist, das effektive Programmieren nie zu erlernen, halten Sie sich an das vorgegebene, klar gegliederte Muster und verfahren bei Ihren Programmen ebenso!

Das Anzapfen von Mikrocomputern

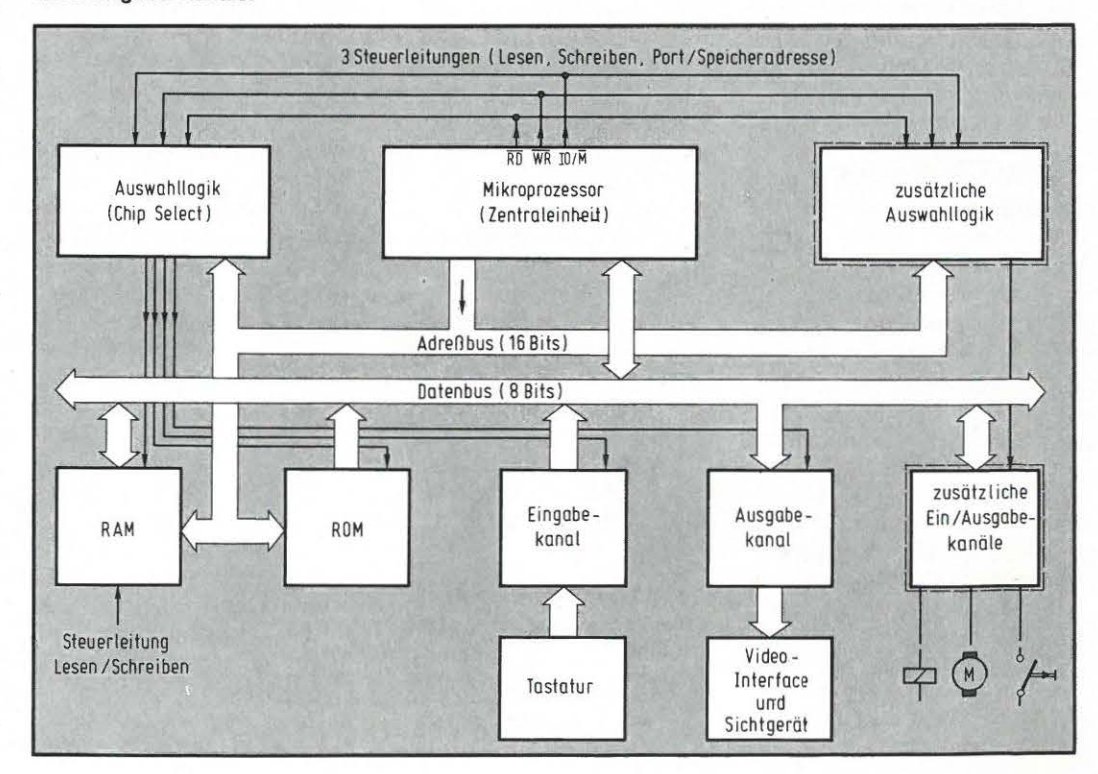
Sicherlich haben Sie beim Spielen mit Ihrem Computer schon öfter den Wunsch gehegt, nicht nur Zeichen auf dem Bildschirm zu produzieren, sondern per Programm auch auf extern angeschlossene Hardware zuzugreifen, also beispielsweise ein Relais einzuschalten oder einen Kontakt abzufragen. Für viele Computer-Enthusiasten bleibt dieser Wunsch unerfüllbar, weil sie ihn nicht in die Tat umsetzen können. Dabei ist der Anschluß zusätzlicher Peripherie viel einfacher als Sie vielleicht glauben.
Das **Bild 115** zeigt den schematischen Aufbau eines Computers. Dieses Schema gilt prinzipiell für alle Computer mit 8-Bit-Datenbus, und dazu gehören nahezu alle Hobby-Geräte des Marktes. Der Mikro**prozessor**, Herzstück des Mikro**computers**, übernimmt neben der eigentlichen Datenverarbeitung („Rechnen") die gesamte Ablaufsteuerung und Koordination der verschiedenen Komponenten wie RAM, ROM und Ein/Ausgabe-Kanäle. Alle diese Komponenten sind an die acht parallelen Leitungen des Datenbusses angeschlossen, über den sich der Informationsfluß zwischen Zentraleinheit (CPU) und umgebenden Schaltungen abspielt. Der Datenbus arbeitet bidirektional, d. h. der Datenfluß auf ihm kann sich zur CPU hin (Lesevorgang) oder von der CPU weg (Schreibvorgang) abspielen.

Da alle Funktionsblöcke im Computer an diesen Datenbus angeschlossen sind, muß der Mikroprozessor als Chef sorgfältig darauf achten, daß hier kein Chaos entsteht. Er tut dies, indem er zu jedem Zeitpunkt ganz genau angibt, mit wem er gerade Daten auszutauschen wünscht. Diese „Zielansprache" geschieht über den Adreßbus, der im allgemeinen 16 Bit breit ist, d. h. er besteht gegenständlich aus 16 Leitungen, die unidirektional, also nur in einer Richtung arbeiten: Nur die CPU gibt im System die Adressen aus, von denen es bei 16 Adreßbits 65 536 verschiedene Möglichkeiten gibt (das sind 64 K und nicht etwa 65 K, weil 1 K als Abkürzung für 1024 steht). Das Ganze müssen Sie sich vorstellen wie ein Riesenpostamt mit 65 536 Postfächern, in die die CPU ihre 8-Bit-Datenbriefe verteilt: Beim Ablegen (Schreibvorgang) wirft sie die Briefchen ins Postfach ein, und beim Abholen (Lesevorgang) entnimmt sie sie dort.

So viel wie ein Adreßbuch

Mit der Adresse allein können die übrigen Schaltungsteile allerdings noch nichts anfangen, und so kommt es, daß die Zentraleinheit in Form der Auswahllogik gewissermaßen einen Sekretär beigestellt bekommt. Diese Logik nimmt sich jede Adresse vor, analysiert sie und folgert dann logisch-messerscharf, zu wem die augenblickliche Adresse gehört. Als Ergebnis wirft diese Logik ein Selektierungssignal aus (engl. **Chip Select**) und leitet es an denjenigen Schaltungsteil weiter, dem die jeweilige Adresse zugeordnet ist. Wer das im Einzelfall ist, das liegt durch die Hardware-Struktur des Computers fest, und das wird beim Schaltungsentwurf bestimmt. Jeder Schaltungsteil im System, der Daten liefern oder entgegennehmen kann, hat eine eigene Adresse; dazu gehört jede einzelne Speicherstelle, in der jeweils ein 8-Bit-Wort abgelegt werden kann, aber natürlich hat auch jeder Ein/Ausgabe-Kanal seine spezifische Adresse, über die er für die CPU erreichbar ist.

Bild 115: Schematischer Aufbau eines Mikrocomputers und Anschlußmöglichkeit zusätzlicher Ein-/Ausgabe-Kanäle.



Es ist einleuchtend, daß in einem Mikrocomputer weit mehr Bedarf an Speicherplatz besteht als an Ein-/Ausgabekanälen, von denen ein paar genügen, während beim Speicher auch einige tausend Stellen zu wenig sein können. Daher ist es wünschenswert, den maximalen Adreßraum von 64 K auch randvoll mit Speicher zu füllen, so daß für die unerläßlichen Ein-/Ausgabe-Kanäle keine freien Adressen mehr übrig bleiben. Aus diesem Grund kann ein ordentlicher Mikroprozessor außer den erwähnten 65 536 Adressen noch weitere erzeugen, die dann speziell für E/A-Kanäle vorgesehen sind. Die Unterscheidung, welcher Typ von Adresse gerade auf dem Adreßbus anliegt, nimmt die CPU mit einem weiteren Signal vor, das so ähnlich wirkt wie ein 17. Adreßbit: Beim 8085 heißt diese Leitung IO/$\overline{M}$ (Abk. f. Input/Output- bzw. Memory-Adresse), und andere Prozessoren verfügen über ähnliche Signale. Der Zustand dieser Steuerleitung (HIGH bzw. LOW) gibt der Auswahllogik an, ob die gerade erzeugte Adresse generell zum Speicher oder zu den E/A-Kanälen gehört. Passend dazu existieren im Befehlssatz Instruktionen, bei deren Ausführung diese Leitung auf LOW liegt, während andere Befehle diese Leitung auf HIGH bringen.

Zwei Richtungsweiser

Wir wollen an dieser Stelle ganz klar festhalten, daß auch die E/A-Kanäle Adressen aus dem 64-K-Basis-Adreßbereich erhalten könnten; sie würden dann programmtechnisch genauso angesprochen wie eine Speicherstelle (über Befehle, bei deren Ausführung die IO/$\overline{M}$-Leitung LOW ist). Die Zuordnung zu Portadressen (IO/$\overline{M}$ auf HIGH) bringt aber den Vorteil, daß der gesamte 64-K-Adreßraum frei bleibt für die Speicherbelegung.
Mit den beiden übrigen Steuerleitungen „Lesen" ($\overline{RD}$) und „Schreiben" ($\overline{WR}$) legt die CPU die Übertragungsrichtung auf dem Datenbus fest. Im ersten Fall erfolgt der Datenfluß zur Zentraleinheit hin, d. h. der Speicher oder ein Eingabekanal geben Daten her; dazu ist nach Ausgabe der Adresse die Leseleitung kurzzeitig aktiv. Beim Schreiben dagegen gibt die CPU Daten an den Speicher oder einen Ausgabekanal ab, wobei die Schreibleitung aktiviert wird. Im Augenblick der Aktivierung von $\overline{RD}$ oder $\overline{WR}$ (natürlich kann immer nur eine von beiden dran sein) sind die Daten auf dem Datenbus stabil, d. h. dann und nur dann muß eine angesprochene Stelle ihre Informationen auf den Datenbus schalten (beim Lesen) bzw. von dort abnehmen (beim Schreiben). Die Selektierungssignale der Auswahllogik nehmen als Basis für ihre Entscheidung, *wer* anzusprechen ist, die ausgegebene Adresse; die eigentliche Aktivierung des betroffenen Bausteins (Chip-Select-Signal) aber erfolgt erst in dem Augenblick, in dem die CPU das $\overline{RD}$- bzw. $\overline{WR}$-Signal ausgibt. Aus Gründen der Störsicherheit sind übrigens sämtliche Selektierungssignale aktiv LOW („negativer" Impuls).
Für einen Eingabekanal (bzw. eine Speicherstelle beim Lesen) ist dieser Impuls die Aufforderung, die im Ruhezustand hochohmigen Ausgänge (Tri-State-Mode) auf den Datenbus zu schalten. Bei einem Ausgabekanal (bzw. einer Speicherstelle beim Schreiben) bedeutet das Aktivierungssignal, daß die Daten vom Datenbus intern zu übernehmen und abzuspeichern sind. Als Ausgabekanal eignet sich bestens der Baustein 74(LS)374 (**Bild 116**), und für die Eingabe muß ein Treiber mit Tri-State-Ausgängen her, z. B. der artverwandte und baugleiche 74(LS)373 (jeder andere Tri-State-Treiber ginge auch, z. B. ein 74(LS)367/68).

Mit Argusaugen an die Adressen

Um die geeigneten Aktivierungssignale zu erzeugen, muß man sich eine vom System nicht benutzte Adresse heraussuchen. Angenommen, es ist die Portadresse E8 noch frei (Portadressen sind im Gegensatz zu Speicheradressen nur 8 Bit lang, daher die zweistellige HEX-Angabe), dann muß eine zusätzliche Auswahllogik diese acht Adreßbits plus die drei Steuerleitungen beobachten und im richtigen Augenblick ein Selektierungssignal erzeugen. Im **Bild 117** erkennen Sie, welchen Zustand die Adreßbits bei „E8" haben; zusätzlich muß IO/$\overline{M}$ auf HIGH sein (Portadresse), und der Ausgabekanal wird bei LOW an $\overline{WR}$, der Eingabekanal dagegen bei LOW an $\overline{RD}$ aktiviert.
Solche Aufgaben der Adreßdecodierung übernimmt liebend gern ein Decoder-IC, z. B. der 74(LS)138 (natürlich könnte man dazu auch Gatter einsetzen). Da ein 138er nur sechs Eingänge besitzt, insgesamt aber 11 Leitungen in die Decodierung einzubeziehen sind (8 Adreßbits plus drei Steuerleitungen), müssen zwei derartige ICs kaskadiert werden (**Bild 118**). An den Ausgängen eines 74138 tut sich nur dann etwas, wenn die beiden Pins 4 und 5 auf LOW und Pin 6 auf HIGH liegt; welcher der acht Ausgänge dann aktiviert wird (auf LOW geht), be-

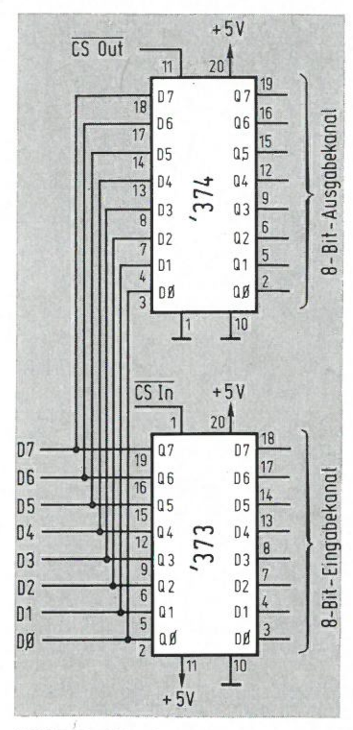


Bild 116: Die ICs für die Ein- und Ausgabe werden an den System-Datenbus angeschlossen und über separate Selektierungssignale aktiviert.

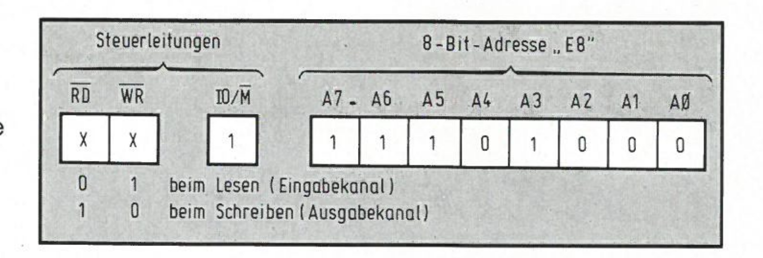


Bild 117: Insgesamt 11 Leitungen sind bei der Decodierung einer Portadresse auf ihren Zustand abzufragen.

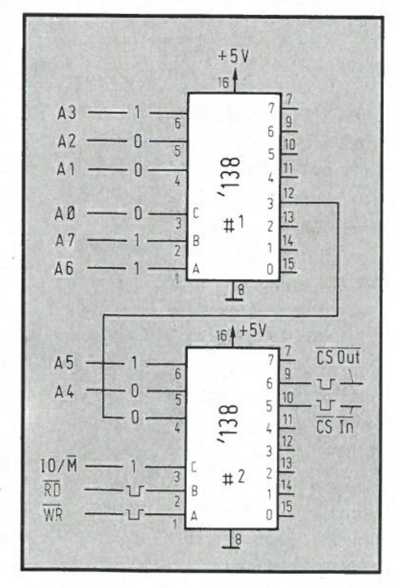


Bild 118: Eines der Freigabesignale für IC 2 stammt vom Ausgang 3 des vorgeschalteten Decoders Nr. 1.

stimmen die drei Bits A, B und C an den Stiften 1, 2 und 3. Die im Bild 118 eingezeichnete Beschaltung gilt wohlgemerkt nur für die beispielhaft angenommene Adresse E8; bei anderen Adressen ergeben sich geänderte Verdrahtungen, wobei sich am Prinzip der CS-Erzeugung aber nichts ändert. Nach diesem Schema lassen sich natürlich auch mehrere Kanäle parallel anschließen, wobei jeder eine eigene Adresse erhalten muß. Darüber hinaus können Sie auf diese Weise auch ganze Baugruppen an einen vorhandenen Mikrocomputer anschließen; Voraussetzung dafür ist nur, daß sich die Zusatz-Peripherie genauso verhält wie die hier vorgestellten '373er/374er: Bei der Ausgabe müssen die Daten mit der positiven Flanke des CS-Signals übernommen werden, und bei der Eingabe müssen die Tri-State-Ausgänge mit dem LOW-Impuls der Auswahllogik auf den Datenbus geschaltet werden.

Die komplette Schaltungserweiterung setzt sich nun aus den beiden in den Bildern 116 und 118 gezeigten Teilen zusammen. Mit dem 8085-Maschinenbefehl „IN E8" gelangen in unserem Beispiel die Pegel der acht Eingangsbits ins A-Register (Akkumulator); bei offenen TTL-Eingängen wäre das das Datenwort „FF" (acht HIGH-Bits). Beim Befehl „OUT E8" wird der Akku-Inhalt an die acht Ausgangsbits übertragen und bleibt dort so lange unverändert erhalten, bis er überschrieben wird; steht im Akku beispielsweise „81" (unterstes und oberstes Datenbit HIGH), dann gehen am Ausgabe-Kanal das Datenbit 0 und 7 auf HIGH, während alle anderen LOW bleiben. Wenn Ihnen die Treiberleistung der TTL-Ausgänge nicht ausreicht, weil Sie beispielsweise ein Relais ein- und ausschalten wollen, dann steuern Sie mit einem Ausgangsbit einen Treibertransistor an, der seinerseits das Relais schaltet.

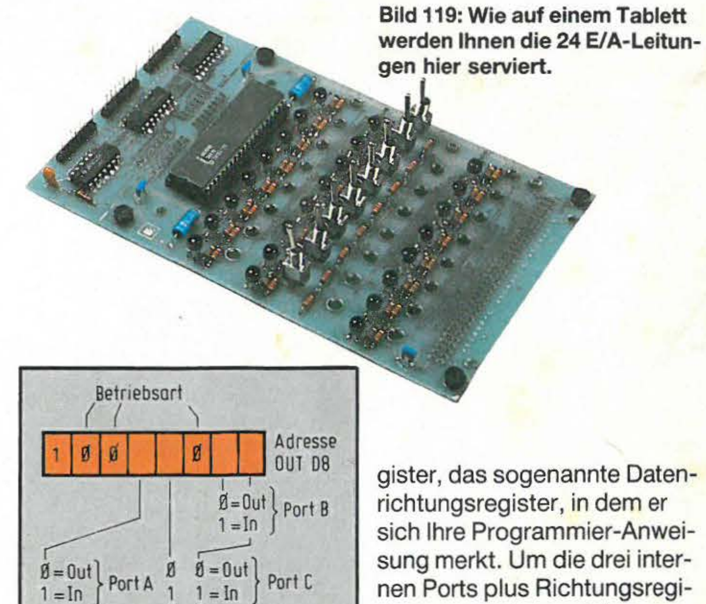


Bild 119: Wie auf einem Tablett werden Ihnen die 24 E/A-Leitungen hier serviert.

Heiße Drähte zum Computer

Der programmierbare Interface-Baustein 8255 erweitert Ihren Computer um 24 bidirektionale Ein-/Ausgabe-Leitungen.

Zur Daten-Ein- und -Ausgabe zwischen Mikrocomputer und umgebender Welt gibt es die verschiedensten Möglichkeiten, beispielsweise den Anschluß von Tri-State-Gattern bzw. D-Flipflops. Ein besonders komfortabler Weg für diesen Datenverkehr ergibt sich beim Einsatz eines programmierbaren Interface-Bausteins, für den so wohlklingende Benennungen im Umlauf sind wie „Peripherer Interface Adapter", abgekürzt PIA, oder auch PIO vom „peripheren I/O" (Input/Output). Wir wollen Ihnen hier den Typen 8255 vorstellen, der aus der 8085-Familie stammt, sich problemlos aber auch an andere Systeme anschließen läßt (**Bild 119**). Dieser Baustein besitzt drei 8 Bit breite Kanäle A, B, und C (auch Ports genannt) für den Datentransfer (zusammen also 24 E/A-Bits), wobei man per Programm die Übertragungsrichtung dieser Ports festlegen kann. Es handelt sich bei diesen Leitungen also nicht um reine Ein- oder Ausgänge, sondern ein und derselbe Anschlußstift kann wahlweise zum Ein- oder zum Ausgang gemacht werden, je nach dem, was Sie ihm sagen (wie Sie ihn polen oder „programmieren"). Dieses Programmieren, d. h. Festlegen der Übertragungsrichtung, muß nach jedem Wiedereinschalten der Versorgungsspannung erneut erfolgen. Das geschieht durch Ausgabe eines sogenannten Steuerwortes an den Baustein, und darin steht codiert die Übertragungsrichtung für die einzelnen Ports (**Bild 120**). Die Darstellung in diesem Bild ist stark vereinfacht; der Baustein „kann" noch wesentlich mehr als nur drei Ports bidirektional versorgen. So ist es beispielsweise möglich, den Port C halbiert anzusprechen, d.h. die eine 4-Bit-Hälfte kann Ausgang sein, während die andere Eingang ist. Die darüber hinaus möglichen Betriebsarten sind für Sonderanwendungen vorgesehen und sollen daher hier unberücksichtigt bleiben. Für das Steuerwort hat so ein E/A-Baustein ein eigenes Register, das sogenannte Datenrichtungsregister, in dem er sich Ihre Programmier-Anweisung merkt. Um die drei internen Ports plus Richtungsregister ansprechen zu können, müssen im System vier Adressen bereitgestellt werden, die der Einfachheit halber aus dem Adreßraum für Portadressen stammen sollten. Bei jeder dieser gewählten Adressen muß der 8255 an seinem $\overline{CS}$-Eingang aktiviert werden. $\overline{RD}$ und $\overline{WR}$ gehen in die Adreßdecodierung selber nicht mit ein, weil der E/A-Baustein selbst das Umschalten der Übertragungsrichtung übernimmt. Er hat dazu zwei entsprechende Eingänge $\overline{RD}$ und $\overline{WR}$ (**Bild 121**). Die beiden Adreßeingänge A0 und A1 am 8255 wählen am Baustein selbst das Ziel an (einen der drei Ports bzw. das Datenrichtungsregister). Davon unabhängig dienen die beiden Decoder-ICs 74(LS)138 dazu, innerhalb des System-Adreßraums die gewählten vier Adressen herauszufischen.

Bild 120: Aufbau des Steuerwortes, das dem 8255 die Übertragungsrichtung der einzelnen Kanäle angibt.

Passend für alle Möpse

Bei der hier gezeigten Schaltung wurde die Adreßdecodierung so gewählt, daß sich für die Ports A, B und C die Adressen A8, B8 und C8 ergeben; das Datenrichtungsregister erhält die Adresse D8, und beim Programmieren entfällt später jegliches Grübeln über die Adreßzuordnung. Die Karte ist zum direkten Anschluß an die 24polige Stiftleiste des MOPPEL-Bus vorgesehen,

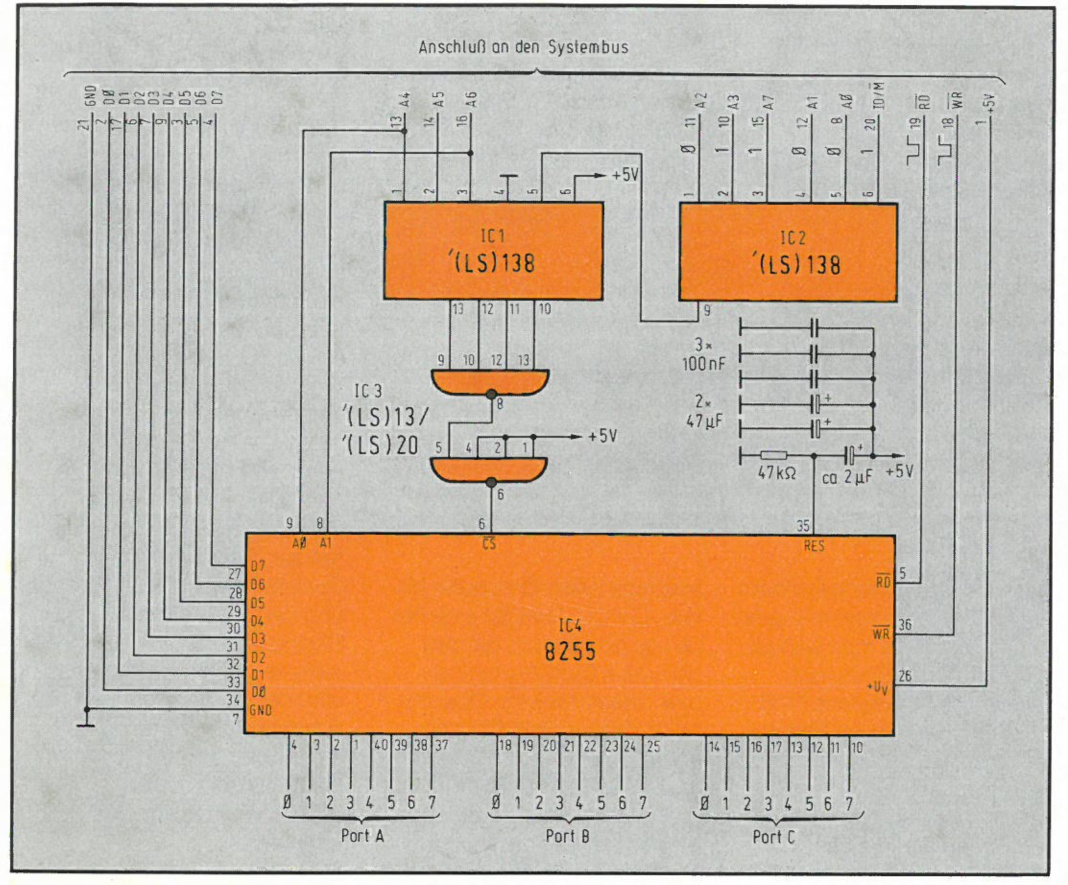


Bild 121: Die beiden Decoder IC1 und IC2 aktivieren den 8255 bei den Portadressen A8, B8, C8 und D8.

ähnlich dem Thermodrucker, aber sie läßt sich natürlich auch an jedem anderen 8085-Bus betreiben. In Systemen, die statt des IO/$\overline{M}$-Signals die beiden Leitungen $\overline{MREQ}$ (Speicherzugriff) und $\overline{IOREQ}$ (Portzugriff) führen, müssen Sie statt IO/$\overline{M}$ die Leitung $\overline{IOREQ}$ anschließen. Dieses Signal muß allerdings aktiv HIGH sein, d.h. es ist in diesem Fall ein Inverter dazwischenzuschalten.

Um die Zustände der 24 Ein-/-Ausgangsbits optisch anzuzeigen, sind Leuchtdioden vorgesehen, die von zusätzlichen Treibertransistoren angesteuert werden (**Bild 122**). Beim Kanal B sind außerdem Schalter vorhanden, mit denen man in der Experimentierphase bestimmte Logikzustände vorgeben kann. **Achtung!** Wenn Sie den Port B als Ausgang programmieren, müssen die Schalter offen sein (Kipphebel nach oben), weil im anderen Fall ein Kurzschluß nach Masse entsteht! Alle 24 E/A-Leitungen sind an den unteren Rand der Platine geführt und können von dort mit den externen Baugruppen verbunden werden. Dabei müssen Sie natürlich darauf achten, daß außer den E/A-Signalen auch die Masseleitung GND an die zusätzliche Peripherie geführt wird.

Bild 122: Treiberschaltung zur Ansteuerung der Leuchtdioden (oben: Ports A und C, unten: Port B).

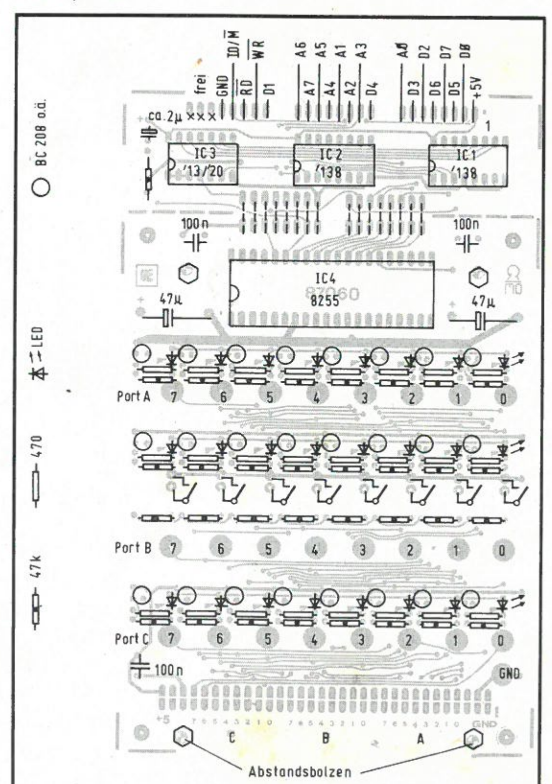
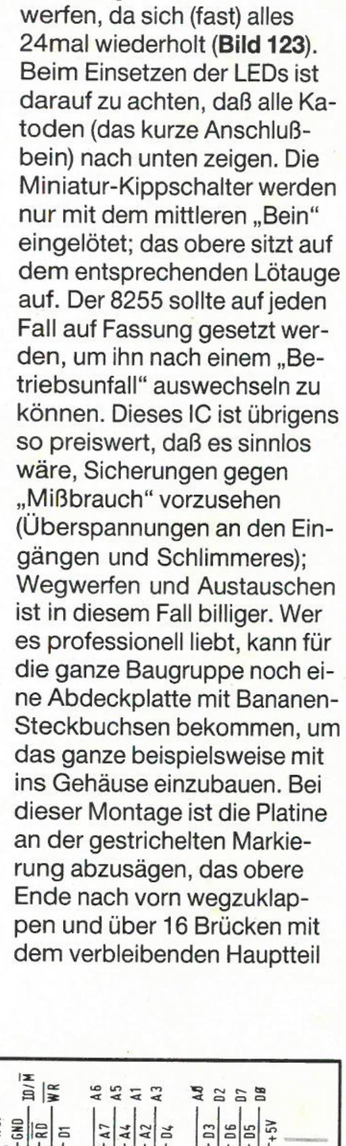


Bild 123: Bestükkungsplan der Platine, die beim losen Betrieb auf vier Plastikfüßchen ruht.



Das Bestücken der Karte dürfte keine großen Probleme aufwerfen, da sich (fast) alles 24mal wiederholt (**Bild 123**). Beim Einsetzen der LEDs ist darauf zu achten, daß alle Katoden (das kurze Anschlußbein) nach unten zeigen. Die Miniatur-Kippschalter werden nur mit dem mittleren „Bein" eingelötet; das obere sitzt auf dem entsprechenden Lötauge auf. Der 8255 sollte auf jeden Fall auf Fassung gesetzt werden, um ihn nach einem „Betriebsunfall" auswechseln zu können. Dieses IC ist übrigens so preiswert, daß es sinnlos wäre, Sicherungen gegen „Mißbrauch" vorzusehen (Überspannungen an den Eingängen und Schlimmeres); Wegwerfen und Austauschen ist in diesem Fall billiger. Wer es professionell liebt, kann für die ganze Baugruppe noch eine Abdeckplatte mit Bananen-Steckbuchsen bekommen, um das ganze beispielsweise mit ins Gehäuse einzubauen. Bei dieser Montage ist die Platine an der gestrichelten Markierung abzusägen, das obere Ende nach vorn wegzuklappen und über 16 Brücken mit dem verbleibenden Hauptteil

wieder zu verbinden (im Bestückungsplan gestrichelt gezeichnet). Um in diesem Fall die Leuchtdioden und Schalter genau fluchtend einlöten zu können, löten Sie diese Bauteile erst dann ein, wenn die Metallplatte auf den vier Abstandsbolzen festgeschraubt ist.

Kanalisierter Transfer

Der Datenverkehr zwischen dem E/A-Baustein und der CPU spielt sich ausschließlich über den Akkumulator ab, d.h. alle Eingaben von außen landen im A-Register und bei Ausgaben müssen die entsprechenden Daten erst in den Akku gebracht werden, ehe der OUT-Befehl aufgerufen werden kann. Um diese E/A-Einheit zu betreiben, müssen Sie, wie oben beschrieben, einmalig das passende Steuerwort laden. Wenn beispielsweise die Ports A und C Ausgangskanäle und Port B ein Eingangskanal sein sollen, müßten Sie „82" ins Datenrichtungsregister (Adresse D8; s.o.) laden: MVI A,82 (Steuerwort 82 in den Akku laden), OUT D8 (Akku-Inhalt 82 an die Portadresse D8 ausgeben, das ist das Datenrichtungsregister im 8255). Danach können Sie beispielsweise die Daten vom Port B in den Akku einlesen (IN B8) und an Port A wieder ausgeben (OUT A8). Wenn Sie aus diesen beiden Befehlen eine Endlosschleife formen, erscheinen an den A-Leuchtdioden immer die an den B-Schaltern eingestellten Zustände. Wer mag, kann am A-Port die B-Daten original und am C-Port invertiert ausgeben (CMA-Befehl, Akku-Inhalt komplementieren), der Phantasie sind da keine Grenzen gesetzt. Die Treiberleistung der 8255-Ausgänge allerdings stößt schnell an ihre Grenzen: Wenn Sie größere Lasten schalten wollen, z. B. Relais, dann gehört ein Treibertransistor davor. Den schließen Sie prinzipiell genauso an wie in Bild 122, nur der Vorwiderstand im Kollektorzweig entfällt. Dafür muß der Transistor bei induktiven Lasten (Erregerspule im Relais) mit einer zur Spule parallel geschalteten „Löschdiode" vor Induktionsspitzen geschützt werden. Der Vollständigkeit halber sei noch erwähnt, daß man einen auf Ausgang geschalteten Port auch lesen kann, d.h. die an seinen Ausgängen liegenden Zustände werden bei einer Leseoperation in den Akku übertragen.

Ein Mikroprozessor unter der Lupe

Wenn ein Computer nach dem Einschalten artig seinen Dienst aufnimmt, macht man sich gewöhnlich kaum Gedanken darüber, welch filigrane Fleißarbeit erforderlich ist, damit ein Befehl nach dem anderen geholt, entschlüsselt und ausgeführt werden kann. Und auch wenn Sie von den internen Zusammenhängen einiges verstehen – Hand aufs Herz: Wissen Sie, welche Anstrengungen ein Mikroprozessor unternehmen muß, um beispielsweise einen Drei-Wort-Befehl auszuführen? Unsere am 8085 aufgehängten Betrachtungen gelten in ähnlicher Form für sämtliche anderen 8-Bit-Prozessoren, die das Herzstück unserer heutigen Heim-Computer bilden. Die CPU besitzt eine Reihe von Eingängen, die den normalen Betrieb stoppen bzw. unterbrechen. Allen voran ist das der **RESET**-Anschluß, bei dessen Aktivierung (u. a.) der Programmzähler auf Null gesetzt wird: resultierend daraus beginnt nach jedem Rücksetzen die Programmausführung bei der Adresse 0000, womit klare Verhältnisse geschaffen werden. – Bei Aktivierung des **HOLD**-Eingangs schaltet die CPU den Daten- und Adreßbus sowie die Steuerleitungen $\overline{RD}$, $\overline{WR}$ und IO/$\overline{M}$ in den hochohmigen Zustand, um einer externen Stelle den Zugriff zum System zu ermöglichen, zum Beispiel beim DMA-Verkehr *(Direct Memory Access,* direkter Speicherzugriff unter Umgehung der CPU). – Die verschiedenen **Interrupt**-Eingänge unterbrechen ein laufendes Programm und veranlassen den Prozessor, an eine bestimmte Stelle im Programmspeicher zu springen; dort ist diejenige Routine abgelegt, die die unterbrechende Stelle bedient, etwa das Einlesen von einer Tastatur, die bei jedem Tastendruck ein Interrupt-Signal auslöst. – Der **READY**-Eingang schließlich dient dazu, das $\overline{RD}$- oder $\overline{WR}$-Signal künstlich zu verlängern; das hat den Sinn, auch langsame Speicher einsetzen zu können und die CPU während deren Reaktionszeit zu bremsen. Bei Aktivierung dieses Eingangs wird der auf dem Daten- und Adreßbus herrschende Zustand „eingefroren", d. h. er bleibt bis zum Abklingen des READY-Signals unverändert erhalten.

Daten und Adressen entwirren

Beim 8085 werden an den Anschlüssen AD0...7 abwechselnd Daten und untere Hälfte jeder Adresse ausgegeben. Um dieses Durcheinander (das natürlich Anschlußstifte spart) wieder zu entwirren,

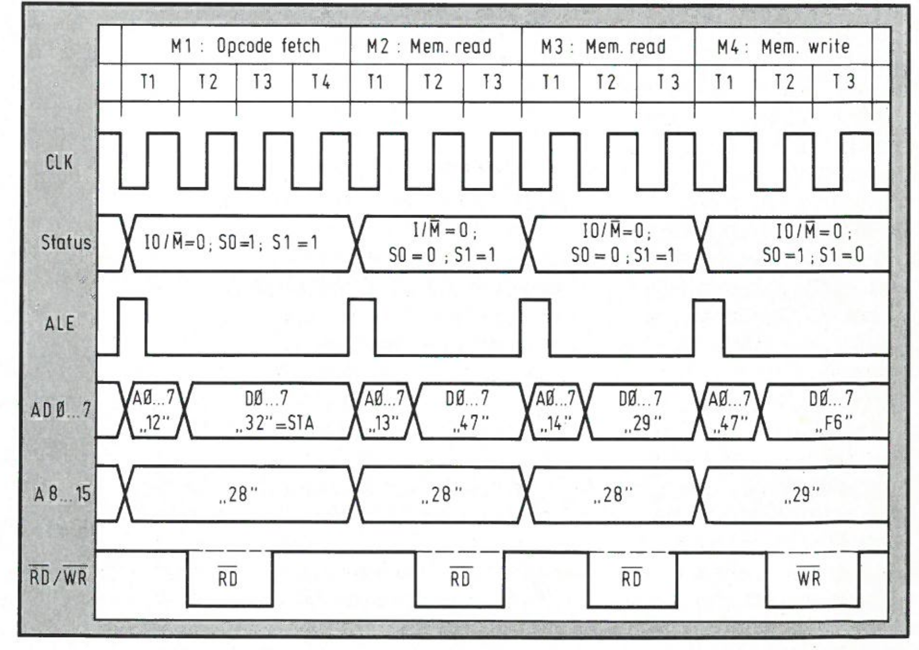



Bild 124: Das Zeitdiagramm des STA-Befehls läßt erahnen, warum die Fehlersuche in einem Mikrocomputer-System ein abenteuerliches Unterfangen wird.

wird die untere Adreßhälfte mit dem CPU-Signal ALE in einen zusätzlich erforderlichen 8-Bit-Speicher überschrieben (Adreß-Latch; in der Regel ein 74(LS)373). Schauen wir der CPU nun einmal auf die Finger, wenn sie sich einen Befehl greift und ausführt. Jeder Befehl gliedert sich in einzelne Phasen, die sogenannten Maschinenzyklen, und jeder Maschinenzyklus setzt sich aus mehreren Taktzyklen zusammen (**Bild 124**). Insgesamt gibt es sieben verschiedene Arten von Maschinenzyklen, und die CPU signalisiert an den Statusleitungen S0, S1 und IO/$\overline{M}$, um welchen Typ es sich bei der gerade begonnenen Operation handelt: Befehl-Hol-Zyklus (*Opcode Fetch;* OF), Speicher lesen (*Memory Read;* MR), Speicher schreiben (*Memory Write;* MW), Port lesen (*IO Read;* IOR), Port schreiben (*IO Write;* IOW), Interrupt stattgeben (*Interrupt Acknowledge;* INA) und Bus in Ruhe (*Bus Idle*). Betrachten wir zur Veranschaulichung einmal folgendes Beispiel: In den Speicherstellen 2812...2814 steht der Drei-Wort-Befehl „STA 2947" (Akkumulator-Inhalt nach Adresse 2947 überschreiben), und bei Erreichen dieses Befehls steht im Akku die Information „F6". Die Ausführung eines jeden Befehls beginnt mit der Hol-Phase OF, d. h. dem Auslesen des Befehlscodes. Dazu gehören die vier Taktzyklen T1...T4, bei denen folgendes passiert: Während T1 wird die Adresse ausgegeben, und zwar die obere Hälfte an A8...A15 und die untere Hälfte an AD0...7. Gleichzeitig erzeugt die CPU den ALE-Impuls, mit dem die untere Adreßhälfte in den erwähnten externen Speicher überschrieben wird. Zusammen mit T2 beginnt die Aktivierung der $\overline{RD}$-Leitung; am Speicher liegt die Adresse 2812 an, und durch den $\overline{RD}$-Impuls erfährt der Speicher, daß die CPU den Inhalt der adressierten Speicherstelle auszulesen wünscht. Im Abschnitt T3 schaltet der Speicher diese Daten (in unserem Beispiel „32") auf den Datenbus, von wo sie direkt ins Befehlshalteregister in der CPU übernommen werden. Nun hat der Prozessor einen Takt lang Zeit (T4), den Befehl zu entschlüsseln und entsprechend zu reagieren. Beim Befehlscode „32" weiß er, daß dies ein Drei-Wort-Befehl ist, daß er also vor der Ausführung erst noch zwei weitere Bytes aus dem Speicher auslesen muß. Dies vollzieht sich in zwei anschließenden Speicher-Lese-Zyklen (Maschinenzyklen M2 und M3), die sich wiederum aus je drei Taktzyklen zusammensetzen: Nächste Adresse 2813 ausgeben, extern abspeichern, Daten auslesen (untere Hälfte der Zieladresse 2947) und dann die folgende Adresse 2814 ausgeben, extern abspeichern und obere Hälfte der Zieladresse 2947 auslesen. Die Zieladresse setzt sich die CPU in zwei internen Hilfsregistern W und Z zusammen, auf die man als Programmierer allerdings keinen Zugriff hat. Erst im folgenden vierten Maschinenzyklus, einer Speicher-Schreib-Operation (MW), wird der Befehl ausgeführt. Das beginnt damit, daß die Zieladresse 2947 aus den Hilfsregistern W und Z auf den Adreßbus geschaltet wird (mit obligatem ALE-Puls wie gehabt) und daß zusammen mit dem $\overline{WR}$-Puls der Inhalt des Akkumulators („F6") auf den Datenbus gelangt. Der Speicher weiß nun, daß er die ankommenden Daten unter der Adresse 2947 abzulegen hat, womit die Befehlsausführung abgeschlossen ist.

Kein berühmter Wirkungsgrad

Wenn Sie mitgezählt haben, sind Sie beim STA-Befehl auf vier Maschinenzyklen mit zusammen 13 Taktpulsen gekommen. Die eigentliche Befehlsausführung, das Einschreiben in den Speicher, nimmt davon gerade zehn Prozent in Anspruch, was ein Verhältnis von 90 % Verwaltung zu 10 % effektiver Arbeit ergibt, und das ist weit schlimmer als in einem verbürokratisierten Verwaltungsapparat!

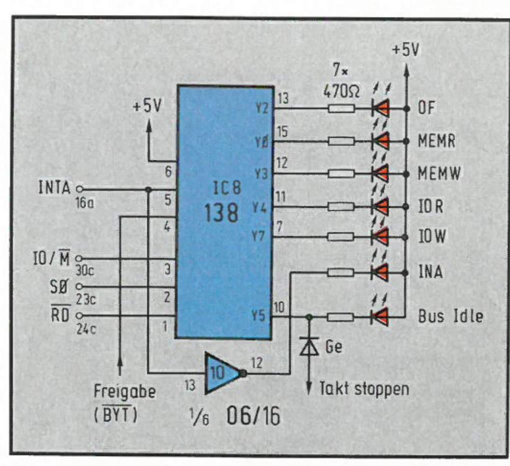


Bild 125: Mit dieser einfachen Schaltung läßt sich der Typ des augenblicklichen Maschinenzyklus anzeigen.

Es gibt im Befehlssatz noch schlimmere, aber auch wesentlich optimalere Beispiele; so wird der MOV-Befehl (Datentransport von einem Register ins andere) in nur vier Taktzyklen ausgeführt, d. h. während T4 wird der Befehl nicht nur entschlüsselt, sondern auch unverzüglich in die Tat umgesetzt. Das geht deshalb so zielstrebig, weil nicht erst Adressen ausgelesen und umständlich zusammengesetzt werden müssen, sondern die Quell- und Zielansprache sind bereits im Befehl selbst enthalten.

Um die Art des augenblicklich laufenden Maschinenzyklus' anzuzeigen (bei der schrittweisen Programmausführung etwa), eignet sich eine Schaltung nach **Bild 125**. Der eingesetzte Decoder wertet Statussignale der CPU aus und steuert entsprechend eine der an den Ausgängen angeschlossenen LEDs an. Mit den eingangs geschilderten Zusammenhängen können Sie sich nun einen Reim darauf machen, wie sich ein Einzelschritt-Betrieb abspielt: Man hält die CPU über den READY-Eingang an und kann sich dann in aller Ruhe den Zustand sämtlicher Daten-, Adreß- und Steuerleitungen betrachten; das Weiterschalten zum nächsten Byte geschieht, indem man einen winzigen Augenblick READY freigibt und die CPU damit den nächsten Maschinenzyklus einläuten kann. Beim folgenden $\overline{RD}$- bzw. $\overline{WR}$-Puls muß READY allerdings wieder aktiv sein, denn während dieser Pulse betrachtet sich der Prozessor seinen READY-Eingang; solange der auf LOW ist, wartet die CPU artig mit weiteren Aktivitäten und gibt uns die Möglichkeit, quasi mit der Lupe ins Computer-Innenleben reinzuschauen.

Schritt für Schritt

Jetzt bringen wir den Mikrocomputer dazu, seine Programme im Einzelschritt- oder Einzelzyklus-Betrieb abzuarbeiten.
Unter dem Begriff „Einzelschritt-Betrieb" ist die Programmausführung Befehl für Befehl zu verstehen, d. h. bei jedem Tastendruck führt die CPU geschlossen einen Befehl aus, egal, ob es sich um einen Ein-Wort- oder Mehr-Wort-Befehl handelt. Beim Einzelzyklus-Betrieb behandelt der Prozessor immer nur den nächsten Maschinenzyklus, d. h. für die Ausführung eines einzigen Befehls können etliche Tastendrücke zum je-

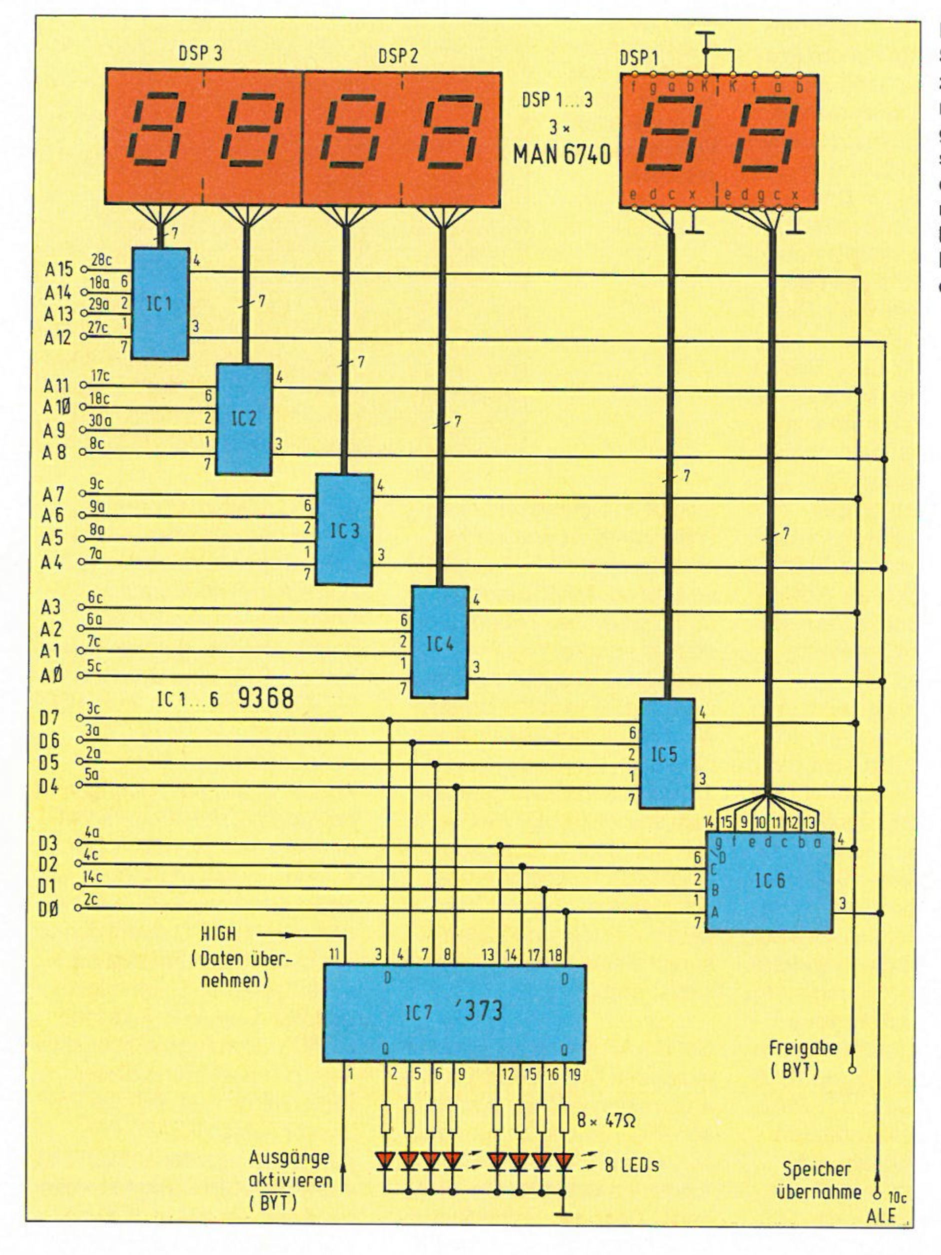


Bild 126: Schaltung zur hexadezimalen Anzeige der Zustände auf dem Daten- und Adreßbus eines 8-Bit-Mikrocomputers.

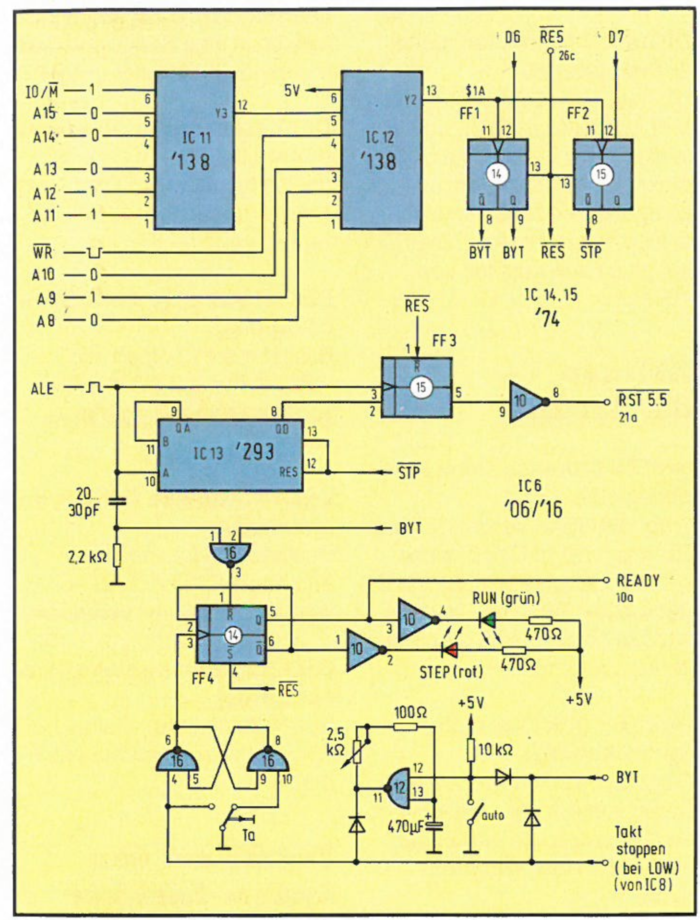
weiligen Weiterschalten erforderlich sein. Beiden Betriebsarten gemeinsam ist die Möglichkeit, nach jedem Fortschalten den Zustand aller Leitungen im System zu inspizieren; dazu gehören außer dem Daten- und Adreßbus z. B. die Anzeige des augenblicklichen Maschinenzyklus' oder die Anzeige aller Registerinhalte, um Fehler aufspüren oder die Befehlsausführung bis ins Detail verfolgen zu können.
Um zunächst einmal die 16 Bits des Adreßbus' und die 8 Bits des Datenbus' hexadezimal darstellen zu können, eignet sich der Schaltungsaufbau nach **Bild 126**; jeweils eine Vier-Bit-Gruppe (ein *Nibble*) vom Daten- und Adreßbus werden dabei an einen hexadezimalen Siebensegment-Decoder (IC1...6) geführt; diese ICs besitzen nicht nur einen internen 4-Bit-Auffangspeicher, sondern die Ausgänge zur Ansteuerung der LED-Anzeigen haben Konstantstromquellen, durch die die sonst üblichen Vorwiderstände entfallen. Mit HIGH-Pegel am Freigabe-Eingang werden die ICs aktiviert, bei LOW bleiben die angeschlossenen Anzeigen dunkel. Mit dem Speicherübernahme-Impuls (positive Flanke) gelangen die eingangsseitig anliegenden Daten in die internen Auffangspeicher, wo sie auch bei LOW am Freigabe-Eingang erhalten bleiben. Um die Daten nicht nur hexadezimal, sondern auch noch als Bitmuster darzustellen, wurde IC7 vorgesehen; dabei handelt es sich um einen 8-Bit-Speicher, bei dem allerdings nur die Tatsache ausgenutzt wird, daß sich seine Ausgänge abschalten, d. h. in den hochohmigen Zustand versetzen lassen (mit HIGH-Pegel am Eingang „Ausgänge aktivieren"). Auf diese Weise läßt sich die gesamte Anzeige-Mimik dunkel tasten, wenn sie nicht gebraucht wird. Durch den Einsatz der hochintegrierten ICs entsteht ferner ein sehr kompakter, platzsparender Aufbau.

Mit gebremstem Schaum

Für jede der beiden o. g. „gebremsten" Betriebsarten gibt es in der Schaltung **Bild 127** ein zugeordnetes Flipflop: FF3 steuert die Einzelschritt- und FF4 die Einzel-Byte-Funktion, und zwar nach gänzlich unterschiedlichen Prinzipien. Jeder Rücksetz-Impuls sperrt diese Flipflops, und zu ihrer Freigabe dienen die beiden assistierenden Flipflops FF1 und FF2. Sie werden über die Decodier-Logik IC11/12 und die Portadresse $1A angesprochen und umgekippt, wenn das Datenwort 80 (gilt für FF2) bzw. 40 (gilt für FF1) nach $1A ausgegeben wird.

Beginnen wir mit dem Einzelzyklus-Betrieb. Der wird eingeleitet durch Ausgabe des Wortes 40 (Datenbit D6 auf HIGH) nach $1A, wodurch FF1 gesetzt wird, d. h. das Signal BYT geht auf HIGH. Damit wird das Gatter am Rücksetz-Eingang von FF4 freigegeben, und der nächste ALE-Impuls führt damit zum Rücksetzen von FF4 (Q-Ausgang auf LOW). Die gerade eingeleitete Schreib- oder Leseoperation wird nun bis in alle Ewigkeit ausgedehnt (sprich: der Prozessor wird angehalten), weil sein READY-Eingang LOW-Pegel aufweist. Erst mit einem Taktimpuls am CLOCK-Eingang von FF4 geht READY wieder auf HIGH, aber nur so lange, bis der nächste ALE-Puls den Q-Ausgang wieder zurücksetzt. Während der Pausen dient die Anzeige-Logik von Bild 126 zur Darstellung der Informationen auf dem Adreß- und Datenbus, und in aller Ruhe kann man sich so die Befehlsausführung bis ins letzte Detail ansehen. Zur Erzeugung des manuellen Taktimpulses dient einmal der Taster (mit nachgeschaltetem Entprell-Flipflop in IC16) und zum anderen der variable Taktgenerator, mit dem in gewissen Geschwindigkeitsgrenzen ein kleckerweiser Automatik-Betrieb erfolgt, den man mit dem Auge noch gut verfolgen kann. Dieser automatische Takt wird nur dann

Bild 127: **Die beiden zentralen Flipflops FF3 und FF4 sind über die Decodier-Logik (oben) erreichbar; die manuelle und automatische Takterzeugung für den Einzel-Byte-Betrieb ist unten erkennbar.**



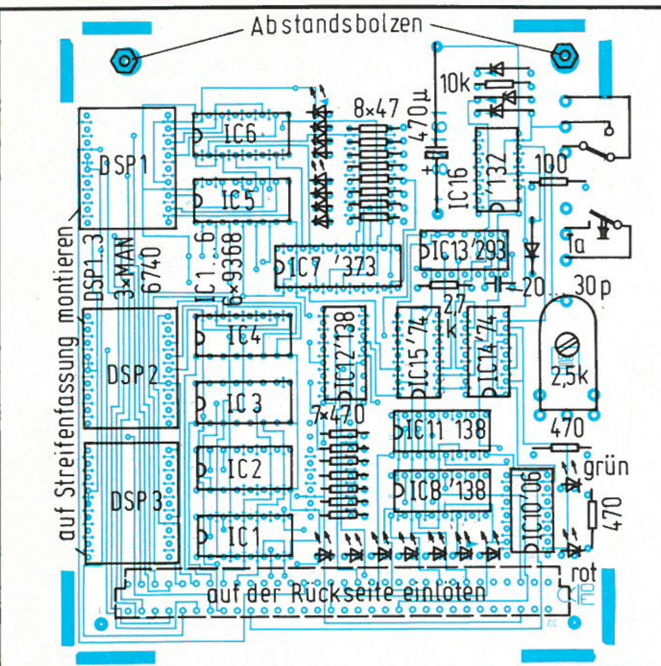


Bild 128: Auf der Platine wird im Einzel-Byte-Betrieb auch der Typ des augenblicklichen Maschinen-Zyklus angezeigt (sieben LEDs links).

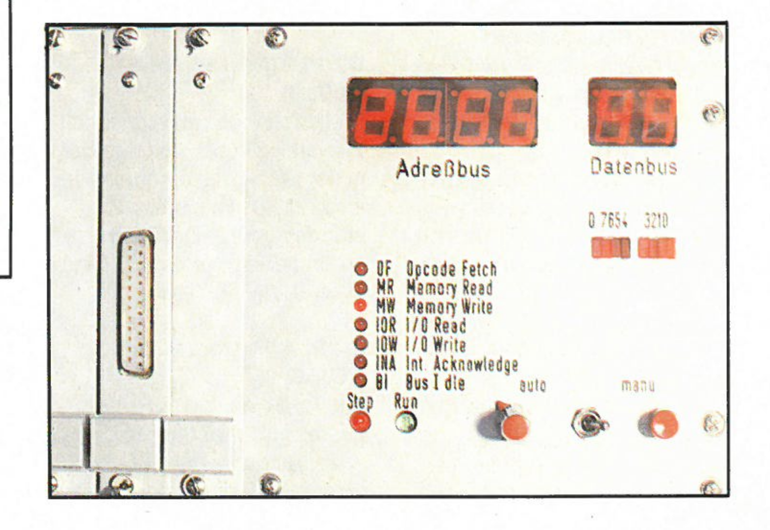
freigegeben, wenn der Schalter „auto" geöffnet ist und am BYT-Ausgang HIGH-Potential anliegt. Die eingezeichneten Leuchtdioden runden den Komfort noch ab: Im Normalbetrieb ohne „Bremse" leuchtet die grüne, und nach erfolgter Einzel-Byte-Aktivierung zeigt die rote den eingeleiteten Warte-Zustand an. Die Signale BYT und $\overline{\text{BYT}}$ werden außerdem noch zur Helltastung der angeschlossenen Siebensegmentanzeigen herangezogen (vgl. Bild 126).

Mit Flip und Flop zum nächsten Befehl

Die Freigabe des Einzelschritt-Flipflops FF3 beginnt mit der Ausgabe des Datenwortes 80 (Datenbit D7 auf HIGH) nach $1A, was den $\overline{\text{Q}}$-Ausgang von FF2 auf LOW bringt und den Zähler IC13 löscht. Dieser Zähler hat die Aufgabe, den RST5.5-Interrupt verzögert auszulösen, und zwar genau acht ALE-Pulse, nachdem $\overline{\text{STP}}$ wieder auf HIGH gebracht worden ist. Mit dieser Verzögerung hat es folgende Bewandtnis: Nach Auslösen des Interrupts am RST5.5-Eingang springt der 8085 auf dem Umweg über die Festadresse 002C in eine Routine, die den Zustand des Daten- und Adreßbusses sowie den Inhalt aller Register auf dem Bildschirm darstellt. Vor Verlassen dieser Interrupt-Service-Routine (die Bestandteil des Monitor-Programms sein muß) sind noch einige Aktivitäten notwendig, ehe der nächste Einzelschritt-Befehl geholt und ausgeführt werden kann; und diese Aktivitäten dauern genau acht ALE-Pulse, die IC13 abzählt und erst mit dem neunten ALE-Puls einen erneuten Interrupt auslöst.
Die gesamte Schaltung läßt sich übersichtlich sortiert auf einem handlichen Platinchen unterbringen (**Bild 128**). Das Format wurde so gewählt, daß sich die Karte in ein vorhandenes 19-Zoll-Gehäuse integrieren läßt, wobei sämtliche Anzeige- und Bedienelemente übersichtlich zugänglich sind (**Bild 129**).

Der Einzelzyklus-Betrieb ist eine reine Hardware-Angelegenheit, d. h. er läßt sich ohne Einschränkungen auf Programme im RAM oder ROM anwenden, indem man zur gewünschten Startadresse springt und dann durch kontinuierliches Tasten einen Maschinenzyklus nach dem anderen durchläuft. Auch der Einzelschritt-Betrieb kann ohne Einschränkungen auf RAM oder ROM angewandt werden, allerdings ist hierfür die entsprechende Interrupt-Service-Routine erforderlich. Mit einer winzigen Feinheit läßt sich dieser Aufbau dahingehend einsetzen, daß ein Echtzeit-Programmlauf bis zu einem bestimmten Haltepunkt (*Breakpoint*) möglich ist, von dem aus der Einzelschritt-Betrieb aufgenommen werden

Bild 129: Zusammen mit der übersichtlich gestalteten Frontplatte läßt sich der Aufbau in ein 19-Zoll-Gehäuse integrieren.



kann. Das allerdings funktioniert nach dem hier beschriebenen Prinzip nur mit Programmen im RAM. Vor dem eigentlichen Programmstart schreibt man in die gewünschte Stoppadresse einen der Einwort-RST-Befehle, bei dessen Ausführung ein Sprung zu einer festen Zieladresse erfolgt; von dort verzweigt man dann in die oben beschriebene Interrupt-Service-Routine für den RST5.5, nachdem der provisorisch eingefügte RST-Befehl wieder durch den ursprünglichen Befehl ersetzt worden ist. Auf diese Weise kann man bestimmte, bereits ausgetestete Programmteile ohne Verzögerungen durchlaufen und erst bei den interessierenden Teilen in den Einzelschritt-Betrieb umschalten.

Der MOPPEL-Profi-Monitor unter der Lupe

Einige der im MOPPEL-Profi-Monitor (EPROM rot) enthaltenen Unterprogramme können Sie in selbst erstellte Programme einbinden, wenn Sie die notwendigen Randbedingungen berücksichtigen. Unerläßlich dafür ist die jeder Monitor-Lieferung beiliegende Symboltabelle, aus der Sie diejenigen Startadressen entnehmen können, die keine Festadressen sind.
Soweit nichts anderslautendes vermerkt ist, können durch den Aufruf dieser Unterprogramme Register-Inhalte verändert werden; wo dies nicht zulässig ist, müssen Sie die betreffenden Register-Inhalte vor dem Unterprogramm-Aufruf per PUSH-Befehl retten und sie nach dem Rücksprung per POP-Befehl zurückholen.
Bei einer Reihe von Unterprogrammen müssen vor dem Aufruf Parameter in Form von Adressen vorgegeben werden (z. B. Start-, End- und Zieladresse beim Kopieren). Diese Parameter-Übergabe erfolgt in den RAM-Zellen SRCBEG (*Source Begin*; Anfangsadresse Quellbuffer), SRCEND (*Source End*; Endadresse Quellbuffer) und DSTBEG (*Destination Begin*; Anfangsadresse Zielbuffer). Hinter diesen drei symbolischen Namen verbergen sich jeweils *zwei* RAM-Zellen (für je eine 16-Bit-Adresse). Mit der Befehlsfolge LXI H,xxyy; SHLD SRCBEG wird die Adresse xxyy unter SRCBEG und SRCBEG+1 abgelegt; der Befehl LHLD SRCBEG lädt H,L mit den unter SRCBEG und SRCBEG+1 abgelegten Daten; die Adressen für SRCBEG, SRCEND und DSTBEG entnehmen Sie bitte Ihrer Symboltabelle.

Ansteuerung der Anzeige

DISPL2 (Festadresse #009D)
Stellt den Akkumulator-Inhalt zweistellig in der Anzeige dar; in der LED-Zeile erscheint der Inhalt des F-Registers (Zustandssignale). Der Rücksprung erfolgt erst nach Betätigen der NXT-Taste.

DISPL4 (Festadresse #00A0)
Stellt den Inhalt des Registerpaares H,L vierstellig in der Anzeige dar; in der LED-Zeile erscheint der Inhalt des F-Registers (Zustandssignale). Der Rücksprung erfolgt erst nach Betätigen der NXT-Taste.

DIGIT1...7 (Adressen siehe Symboltabelle)
Überschreibt 1...7 Stellen des Display-Buffers 2FC5...2FCB in die Digits 1...7 der Anzeige; dieses Unterprogramm erzeugt die 1-kHz-Multiplexfrequenz für die Anzeige. DIGIT7 hat eine feste Laufzeit von 10 ms.

Abfrage der Tastatur

HHKEY (Adresse siehe Symboltabelle)
Fragt die HEX-Tasten ab; beim Rücksprung ist CY=0, wenn keine Taste gedrückt war. Bei gedrückter Taste ist CY=1, und die Tasten-Nr. 0...F steht im Akkumulator.

HEXKEY (Adresse siehe Symboltabelle)
Fragt die HEX-Tasten ab, liest Daten von dort ein und schiebt sie (Dezimalpunkt-abhängig) ins Daten- oder Adreßfeld nach.

RELSH (Adresse siehe Symboltabelle)
Wartet auf das Loslassen einer HEX-Taste und aktiviert während dieser Zeit die Anzeige.

CCKEY (Adresse siehe Symboltabelle)
Fragt die Befehlstasten ab; beim Rücksprung ist CY=0, wenn keine Taste gedrückt war. Bei gedrückter Taste ist CY=1, und die Tasten-Nr. 0...6 (für NXT, BST, FCT, REG, ADR, DAT, RUN) steht im Akkumulator.

RELSC (Adresse siehe Symboltabelle)
Wartet auf das Loslassen einer Befehlstaste und aktiviert während dieser Zeit die Anzeige.

Erzeugen fester Laufzeiten

Beim Aufruf dieser Unterprogramme bleibt die Anzeige dunkel; für Verzögerungen mit aktiver Anzeige können Sie z. B. auf DIGIT7 (s. o.) zurückgreifen und dessen Basis-Laufzeit von 10 ms durch Einkleiden in eine Schleife vervielfachen (vgl. Beispiel unten).

DELY1 (Adresse siehe Symboltabelle)
Besitzt feste Laufzeit von 1 ms; Register werden nicht verändert.

DELYB (Adresse siehe Symboltabelle)
Besitzt feste Laufzeit von <B>*1 ms; mit Ausnahme von B wird kein Register verändert.

DELAY (Adresse siehe Symboltabelle)
Besitzt feste Laufzeit von 100 ms; mit Ausnahme von B wird kein Register verändert.

ONSEC (Adresse siehe Symboltabelle)
Besitzt feste Laufzeit von 1 s; Register werden nicht verändert.

Zugriff auf den System-Speicher

FILL (Festadresse #0085)
Füllt einen Speicherbereich mit einer Konstanten; vor dem Aufruf folgende Parameter laden (vgl. o.): SRCBEG=Anfangsadresse, SRCEND=Endadresse, DSTBEG=Konstante.

COPY (Festadresse #0088)
Verschiebt einen Speicherbereich (auch überlappend); vor dem Aufruf folgende Parameter laden (vgl. o.): SRCBEG= Anfangsadresse Quellbereich, SRCEND=Endadresse Quellbereich, DSTBEG=Anfangsadresse Zielbereich.

MEMCHK (Festadresse #005B)
Ermittelt auf der großen Speicherkarte die höchste vorhandene RAM-Adresse und übergibt diese in den Registern A und B (obere Hälfte in A).

COMPAR (Adresse siehe Symboltabelle)
Vergleicht die Registerpaare D,E und H,L; bei Gleichheit ist nach dem Rücksprung Z=1.

UHR (Festadresse #0070)
Zählt den Uhrzeit-Buffer UHRBUF (2F7D...2F8A) um eine Sekunde weiter; die Übertragsbildung berücksichtigt dabei sowohl den Wochentag als auch die unterschiedliche Monatslänge (einschließlich Schaltjahre).

Ansteuerung peripherer Baugruppen

CASOT (Festadresse #006A)
Ausgabe über das langsame Kassetten-Interface (auf die CPU aufgesteckt) mit 250 Baud Übertragungsrate; vor dem Aufruf Parameter laden: H,L=Anfangsadresse Quellbereich; D,E=Blocklänge.

CASIN (Festadresse #006D)
Eingabe über das langsame Kassetten-Interface (auf die CPU aufgesteckt) mit 250 Baud Übertragungsrate; vor dem Aufruf Parameter laden: H,L=Anfangsadresse Zielbereich;
D,E=Blocklänge.

WRITUR (Festadresse #0079)
Stellt die Echtzeit-Uhr; vor dem Aufruf Parameter laden: H,L=Stunden,Minuten; B,C=Wochentag (1...7; 1=Montag), Kalendertag; D,E=Monat,Jahr.

READUR (Festadresse #007C)
Liest die Echtzeit-Uhr und überschreibt die BCD-Daten ins RAM, beginnend bei der beim Einsprung in H,L stehenden Adresse.

TI (Festadresse #0061)
Liest die Echtzeit-Uhr und setzt die Informationen in den ASCII-Code um, beginnend bei 2FE0 (Format: Di,01.02.83; 09:15:37h).

PRTHEX (Festadresse #007F)
Steuert den Thermodrucker an (hexadezimaler Speicherabzug); vor dem Aufruf Parameter laden: H,L=Anfangsadresse; D,E=Endadresse.

PRTASC (Festadresse #0082)
Steuert den Thermodrucker an (Klartext-Ausdruck); vor dem Aufruf Parameter laden: H,L=Anfangsadresse Textbuffer; SRCEND-Zeilenzahl.

WRPROM (Festadresse #0073)
Steuert den EPROM-Programmierzusatz an (Programmieren von EPROMs); vor dem Aufruf Parameter laden: A=EPROM-Typ (16/32); H,L=Anfangsadresse Quellbereich; D,E=Endadresse Quellbereich; B,C=Anfangsadresse im EPROM.

RDPROM (Festadresse #0076)
Steuert den EPROM-Programmierzusatz an (Lesen von EPROMs); vor dem Aufruf Parameter laden: A=EPROM-Typ (16/32); H,L=Anfangsadresse EPROM; D,E=Endadresse EPROM; B,C=Anfangsadresse Zielbereich.

V24OUT (Festadresse #0097)
Gibt Daten über die serielle CPU-Schnittstelle aus; vor dem Aufruf Parameter laden: H,L=Startadresse Quellbereich; D,E=Endadresse Quellbereich; B,C=Baudrate (110, 300, 600, 1200, 1800, 2400, 4800, 9600).

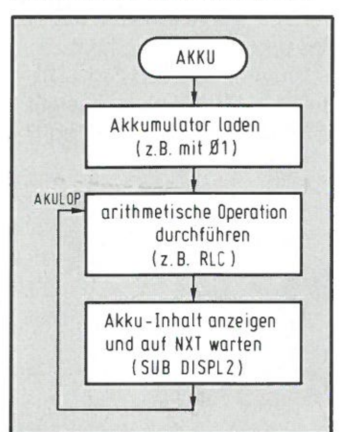


Bild 130: Das Unterprogramm DISPL2 stellt den Inhalt der Register A und F dar und wartet auf die Betätigung der NXT-Taste.

V24IN (Festadresse #009A)
Liest Daten über die serielle CPU-Schnittstelle ein; vor dem Aufruf Parameter laden: H,L=Anfangsadresse Zielbereich; D,E=Endadresse Zielbereich; B,C=Baudrate (110, 300, 600, 1200, 1800, 2400, 4800, 9600).

Reservierte RAM-Bereiche (Adressen siehe Symboltabelle)

Register-Buffer REGA...REGP
Nach einem Hardware-RESET und im Einzelschritt-Betrieb werden die Register-Inhalte in folgende RAM-Zellen gerettet: REGA, REGB, REGC, REGD, REGE, REGF, REGH, REGL, REGS (Stack-Pointer, upper) und REGP (Stack-Pointer, lower).

Interrupt-Sprungverteiler
Diejenigen Sprungziele, zu denen bei externen Programmunterbrechungen (Interrupts) verzweigt werden soll, sind unter folgenden symbolischen Namen im RAM-Sprungverteiler abzulegen: TRAP, RST3, RST4, RST5, RST55, RST6, RST65, RST7, RST75. Die Ziele RST0, RST1 und RST2 sind vom System intern belegt.

Display-Buffer
In den RAM-Zellen DIG1 bis DIG7 stehen diejenigen Informationen, die von den Unterprogrammen DIGIT1...7 in die Siebensegmentanzeigen überschrieben werden.

Portadressen der HEX-Tastatur
Die Siebensegmentanzeigen DIGIT1...7 belegen die Portadressen 10...70, und der LED-Zeile ist die Portadresse 00 zugeordnet. Die Tastatur ist in Form einer 3 × 8-Matrix organisiert, deren Zeilenleitungen folgende Portadressen besitzen: EC (HEX-Tasten 0...7), DC (HEX-Tasten 8...F), BC (Befehlstasten).

Programmbeispiel Ergebnisanzeige

Unter Verwendung der oben beschriebenen Unterprogramme soll eine arithmetische Operation mit anschließender Anzeige des Ergebnisses durchgeführt werden (**Bild 130**). Erfolgt die Operation im Akkumulator, kann unmittelbar das Unterprogramm DISPL2 (s. o.) zur Ergebnisanzeige aufgerufen werden; da dieses Unterprogramm gleichzeitig die NXT-Taste abfragt, läßt sich mit einem anschließenden Sprungbefehl eine Endlosschleife aufbauen, die nach jedem Druck auf NXT einmal durchlaufen wird und den dabei modifizierten Akkumulator-Inhalt darstellt (**Bild 131**). Setzen Sie unterschiedliche Verarbeitungsbefehle ein (Speicherzelle 2803; z. B. INR A=Akkumulator um Eins erhöhen; DCR A=Akkumulator um Eins erniedrigen; RLC=Akkumulator-Inhalt links verschieben; ADD B= zum Akkumulator-Inhalt <Reg B> addieren usf.) und verfolgen Sie deren Auswirkung in der Anzeige!

Programmbeispiel Laufschrift

Mit geringem Aufwand lassen sich die Monitor-Unterpro-

```
2800                      ORG    2800H

2800 3E 01      AKKU      MVI    A,01       EIN BIT AUF HIGH SETZEN
2802 07         AKULOP    RLC               AKKU ZYKLISCH LINKS VERSCHIEBEN
2803 CD 9D 00             CALL   009DH      UP DISPL2
2806 C3 02 28             JMP    AKULOP     ENDLOSSCHLEIFE
```

Bild 131: Das Assembler- und Maschinenprogramm zu Bild 1.

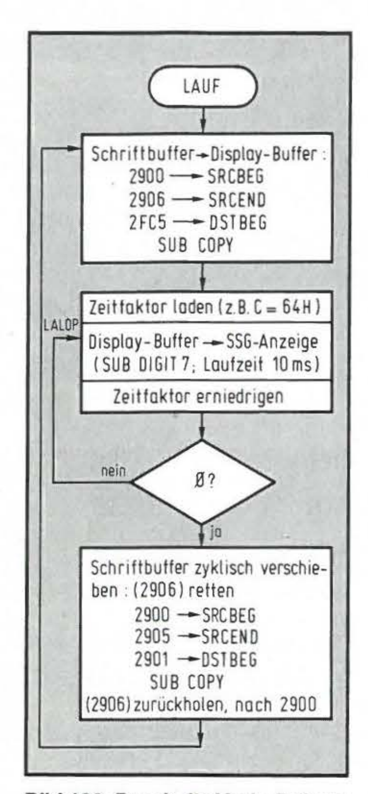


Bild 132: Durch die Verknüpfung zweier Monitor-Unterprogramme entsteht in der Siebensegmentanzeige eine kleine Laufschrift.

```
2800                      ORG   2800H
                    *
                    *     SCHRIFTBUFFER IN DISPLAY-BUFFER UEBERSCHREIBEN
                    *
2800 21 00 29  LAUF       LXI   H,2900H   ANFANG QUELLBUFFER
2803 22 D1 2F             SHLD  2FD1H     NACH SRCBEG
2806 21 06 29             LXI   H,2906H   ENDE QUELLBUFFER
2809 22 D3 2F             SHLD  2FD3H     NACH SRCEND
280C 21 C5 2F             LXI   H,2FC5H   ANFANG ZIELBUFFER
280F 22 D5 2F             SHLD  2FD5H     NACH DSTBEG
2812 CD 88 00             CALL  0088H     UP COPY
                    *
                    *     DISPLAY-BUFFER IN ANZEIGE UEBERSCHREIBEN
                    *
2815 0E 64                MVI   C,100     100*10 ms LAUFZEIT
2817 CD 4F 02  LALOP      CALL  024FH     UP DIGIT7 (10 ms)
281A 0D                   DCR   C         ZEITFAKTOR ERNIEDRIGEN
281B C2 17 28             JNZ   LALOP     NOCH NICHT NULL: JMP
                    *
                    *     SCHRIFTBUFFER ZYKLISCH VERSCHIEBEN
                    *
281E 3A 06 29             LDA   2906H     SCHRIFTBUFFER (MAX) HOLEN
2821 F5                   PUSH  PSW       UND IM STACK RETTEN
2822 21 00 29             LXI   H,2900H   ANFANG QUELLBUFFER
2825 22 D1 2F             SHLD  2FD1H     NACH SRCBEG
2828 21 05 29             LXI   H,2905H   ENDE QUELLBUFFER
282B 22 D3 2F             SHLD  2FD3H     NACH SRCEND
282E 21 01 29             LXI   H,2901H   ANFANG ZIELBUFFER
2831 22 D5 2F             SHLD  2FD5H     NACH DSTBEG
2834 CD 88 00             CALL  0088H     UP COPY
2837 F1                   POP   PSW       ALTEN INHALT (2906) HOLEN
2838 32 00 29             STA   2900H     UND NACH SCHRIFTBUFFER (MIN)
283B C3 00 28             JMP   LAUF
```

Bild 133: Das Assembler- und Maschinenprogramm zu Bild 3.

gramme dazu verwenden, in der Siebensegmentanzeige eine kleine Laufschrift zu erzeugen. Dazu wird im RAM-Bereich 2900 bis 2905 ein Schriftbuffer definiert, in dem die darzustellenden Schriftsymbole abgelegt werden. Das eigentliche Programm setzt sich dann aus nur drei Aktivitäten zusammen (**Bild 132**): 1. Übertragen des Schriftbuffers in den Display-Buffer (Einsatz des Unterprogramms COPY); 2. Überschreiben des Display-Buffers in die Anzeige (Unterprogramm DIGIT7, eingekleidet in eine Schleife, um gleichzeitig eine Verzögerung zu erzeugen); 3. Verschieben des Schrift-Buffers um eine Stelle (wiederum mit Hilfe von COPY). Laden Sie das Programm gemäß **Bild 133** ins RAM und starten Sie es per RUN; mit dem Multiplikator in Adresse 2816 bestimmen Sie die Anzahl der Durchläufe von DIGIT7 und damit die Geschwindigkeit der Rotationsbewegung. Bei der Gestaltung der Schriftsymbole berücksichtigen Sie bitte, daß jedes Bit im Display-Buffer mit einem Segment der Anzeige korrespondiert: Bit 0 mit Segment a usf. bis Bit 7 mit Segment f; um das betreffende Segment zu aktivieren, muß das zugehörige Bit auf HIGH sein (Beispiel in Bild 133: Schriftbuffer ab 2900).

Der MOPPEL-Video-Monitor unter der Lupe

Der MOPPEL-Video-Monitor (EPROM gelb) erweitert den Profi-Monitor (EPROM rot) dahingehend, daß ASCII-Tastatur und Bildschirm angesteuert werden können. In diesem gelben EPROM ist eine Reihe von Unterprogrammen enthalten, die man in eigene Programme einbinden kann, um sich die zeitaufwendige Neuprogrammierung zu ersparen.

Soweit bei der Vorstellung der Monitor-Unterprogramme keine Festadressen angegeben sind, entnehmen Sie die Anfangsadresse bitte der jeder Lieferung beiliegenden Symboltabelle.

Alle hier genannten Unterprogramme setzen voraus, daß das System mit ASCII-Tastatur und Video-Interface (sowie mit den EPROMs rot *und* gelb) bestückt ist. Selbstverständlich können dabei auch diejenigen Unterprogramme noch aufgerufen werden, die im Zusammenhang mit dem roten Monitor vorgestellt wurden; lediglich die Routinen zur Ansteuerung von HEX-Tastatur und Siebensegmentanzeige sind beim Bildschirmbetrieb aus naheliegenden Gründen nicht mehr aktivierbar. Die Version des Video-Monitors steht übrigens in Adresse 1048 („71" an dieser Stelle bedeutet z. B. Video-Monitor, Ver. 1).

Achtung! Es kommt vor, daß sich die hier angegebene Anfangsadresse für ein Unterprogramm von der in der Symboltabelle genannten unterscheidet. Dies ist nicht etwa ein Widerspruch, sondern es bedeutet, daß bei der erstgenannten Adresse ein Sprung zur zweiten Adresse steht; diese Organisationsform ist für modular erweiterbare Software-Pakete unerläßlich.

Konsole rein, Konsole raus

Im Prinzip macht der Video-Monitor eigentlich nur zwei Dinge, und das sind die Abfrage der ASCII-Tastatur und die Bildschirmdarstellung des jeweils eingegebenen Zeichens. Das können Sie unmittelbar verfolgen, wenn Sie Tasten betätigen und die entsprechenden Zeichen hintereinander auf dem Schirm wiederfinden. Die fest im Computer installierten Ein-/Ausgabe-Einheiten Tastatur und Bildschirm nennt man fachmännisch „Konsole" (engl. *Console*).
Folgende drei Unterprogramme verwalten diese zweigeteilte (Ein- und Ausgabeteil) Konsole: **CI** (Console Input = Zeichen von der Tastatur einlesen), **CSTS** (Console Status = Status der Tastatur ermitteln, d. h. Abfrage, ob eine Taste gedrückt ist) und **CO** (Console Output = Zeichen auf dem Bildschirm ausgeben).
Den überwiegenden Teil seines Lebens verbringt der Video-Monitor nun damit, diese Unterprogramme in einer Endlosschleife zu durchlaufen und hin und wieder zu anderen Programmteilen zu verzweigen. Wie Sie wissen, erfolgt diese Verzweigung immer dann, wenn Sie nach Eingabe eines der 26 Kommandos A...Z die Taste Return betätigen; das ist für den Monitor der dezente Hinweis nachzusehen, welches Kommando Sie hinter der Bildschirmzeile **Anweisung an MOPPEL >** eingegeben haben, und der entsprechende Sprung in das zugehörige Unterprogramm ist dann eine reine Formsache.
Der Video-Monitor ist aber noch ein bißchen pfiffiger als eben dargestellt; er erkundigt sich nämlich fortlaufend beim Video-Interface, ob Sie als Anwender das kleine Darstellungsformat (18 Zeilen mal 40 Zeichen) oder das große Format (24 × 80 Zeichen) eingestellt haben; Sie erinnern sich, daß dazu auf der Video-Interface-Karte die Brücken 3/5 bzw. 4/6 vorgesehen sind. Je nach gewählter Einstellung verhält sich der Video-Monitor unterschiedlich, denn der muß ja beispielsweise wissen, ob er seine „Anweisung an MOPPEL >" in die Textzeile Nr. 18 (kleines Format) oder in Zeile 24 schreiben muß.
Wenn Sie die Beschreibung des Video-Interfaces aufmerksam gelesen haben, dann wissen Sie, daß der Bildschirm fortlaufend mit denjenigen Daten vollgeschrieben wird, die (im ASCII-Code) im Video-RAM stehen (ab Adresse 3000). Das sieht so aus, daß die in 3000 stehende Information links oben auf dem Bildschirm erscheint, die aus 3001 steht daneben und so fort.
Wollen Sie zum Beispiel etwas linksbündig in die zweite Textzeile schreiben, dann hängt die Adresse für die zugehörige RAM-Zelle vom gewählten Bildformat ab; beim kleinen Format ist sie um dezimal 40 Plätze (= 28 HEX) vom Anfang versetzt (lautet also 3028), während sie beim Großformat 3050 lautet (80D = 50 HEX-Stellen vom Anfang 3000 versetzt). Sie können also direkt auf jeden beliebigen Bildschirmplatz schreiben, indem Sie das gewünschte ASCII-Zeichen per STA-Befehl in den Bildspeicher ab 3000ff. schreiben.

Übrigens lädt der Video-Monitor nach jedem RESET (unabhängig vom gewählten Bildformat) diejenige Adresse nach 3784/85, die dem Bildschirmplatz links unten entspricht (Beginn der untersten Textzeile). Wenn Sie dorthin z. B. einen Text bringen wollen, brauchen Sie nicht umständlich die Zieladresse auszurechnen, sondern es genügt, das Registerpaar H,L mit dem Inhalt der RAM-Zellen 3784/85 zu laden (LHLD-Befehl).
Bei einer Ausgabe über das Unterprogramm CO gelangt das betreffende Zeichen an diejenige Stelle, an der gerade der Cursor blinkt. Der Video-Monitor erfährt diese Position, indem er die RAM-Zellen 3782/83 ausliest (genannt UPDATE). Mit Hilfe der hier beschriebenen Unterprogramme haben Sie die Möglichkeit, den Cursor an jede beliebige Stelle auf dem Schirm zu verpflanzen (zu positionieren).

Ansteuerung des Bildschirms

CO (Festadresse #0049)
Gibt das in Register C enthaltene ASCII-Zeichen auf dem Bildschirm aus; die Bildschirmposition wird dabei durch die in 3782/83 stehende Adresse UPDATE festgelegt.

ADROT (Festadresse #103C)
Gibt die im Registerpaar D,E enthaltenen Daten vierstellig (hexadezimal) auf dem Bildschirm aus; die Bildschirmposition wird dabei durch die im Registerpaar H,L stehende Adresse festgelegt.

BYTOT (Festadresse #103F)
Gibt die im Register A (Akkumulator) enthaltenen Daten zweistellig (hexadezimal) auf dem Bildschirm aus; die Bildschirmposition wird dabei durch die im Registerpaar H,L stehende Adresse festgelegt.

BYTBIN (Festadresse #1042)
Stellt die im Register A (Akkumulator) enthaltenen Daten als achtstelliges Bitmuster (Folge aus 0 und 1) auf dem Bildschirm dar; die Bildschirmposition wird dabei durch die im Registerpaar H,L stehende Adresse festgelegt.

VIDOT (Adresse siehe Symboltabelle)
Gibt eine Folge von ASCII-Zeichen (die durch „00" abzuschließen ist) auf dem Bildschirm aus; die Anfangsadresse der Zeichenkette steht im Registerpaar H,L, und die Bildschirmposition wird durch die augenblickliche Cursor-Position festgelegt.

STRING (Festadresse #1021)
Wie VIDOT, jedoch beginnt die Ausgabe beim Anfang der nächsten Bildzeile.

UNDRLN (Festadresse #102A)
Wie STRING, jedoch erfolgt die Ausgabe mit Inversdarstellung (dunkle Schrift auf hellem Hintergrund).

SAMPLE (Festadresse #101B)
Gibt den gesamten Zeichenvorrat auf dem Bildschirm aus (vgl. Bild 79).

Abfrage der ASCII-Tastatur

CI (Festadresse #0043)
Fragt die ASCII-Tastatur ab und übergibt bei gedrückter Taste das entsprechende ASCII-Zeichen im Register A (Akkumulator); der Rücksprung aus diesem Unterprogramm erfolgt erst, nachdem eine Taste betätigt worden ist (aber ohne auf das Loslassen zu warten).

CSTS (Festadresse #0052)
Ermittelt den Status der ASCII-Tastatur. Ist eine Taste gedrückt, so ist beim Rücksprung das Zero-Flag = 0; andernfalls ist Z = 1.

Cursor-Steuerung

CRSON (Festadresse #1036)
Schaltet den (blinkenden) Cursor ein.

CRSOFF (Festadresse #1039)
Schaltet den Cursor aus.
Achtung! Beim Sprung in ein Anwender-Programm (mittels G-Kommando) wird der Cursor automatisch ausgeschaltet.

CRSPOS (Festadresse #1033)
Positioniert den Cursor an diejenig Stele auf dem Bildschirm, die der im Registerpaar H,L stehenden Video-RAM-Adresse entspricht.

HOME (Adresse siehe Symboltabelle)
Bringt den Cursor in die linke obere Ecke des Bildschirms.

DOWN (Adresse siehe Symboltabelle)
Bringt den Cursor unabhängig vom gewählten Bildformat um eine Textzeile nach unten; bei Überlauf geht es in der obersten Zeile weiter.

RIGHT (Adresse siehe Symboltabelle)
Bringt den Cursor unabhängig vom gewählten Bildformat um eine Stelle nach rechts; bei Überlauf geht es in der nächsten Textzeile weiter.

UP (Adresse siehe Symboltabelle)
Bringt den Cursor unabhängig vom gewählten Bildformat um eine Textzeile nach oben; bei Überlauf geht es in der untersten Zeile weiter.

LEFT (Adresse siehe Symboltabelle)
Bringt den Cursor unabhängig vom gewählten Bildformat um eine Stelle nach links; bei Überlauf geht es in der vorherigen Textzeile weiter.

CARET (Adresse siehe Symboltabelle)
Bringt den Cursor unabhängig vom gewählten Bildformat zum Anfang der untersten Textzeile.

LINFED (Adresse siehe Symboltabelle)
Bewirkt einen Zeilenvorschub auf dem Bildschirm; die oberste Textzeile geht dabei verloren, und als unterste wird eine Leerzeile angehängt.

CRLF (Adresse siehe Symboltabelle)
Führt die beiden Funktionen CARET und LINFED aus.

Zugriff auf den Video-Speicher

CLRLIN (Festadresse #1030)
Löscht unabhängig vom gewählten Bildformat die unterste Textzeile.

CLRVID (Festadresse #102D)
Löscht den auf dem Schirm erscheinenden Teil des Video-RAMs von 3000...377F; die z. T. vom Video-Monitor belegten übrigen Zellen bis 3FFF bleiben davon unberührt.

CLRAM (Adresse siehe Symboltabelle)
Löscht das gesamte Video-RAM von 3000...3FFF einschließlich der z. T. vom Monitor belegten Hilfszellen 3780...37FF.

Ansteuerung peripherer Baugruppen

PO (Festadresse #004C)
Gibt die im Register C stehenden Daten (1 Byte) über das schnelle Kassetten-Interface aus (Anschluß an der TB-Buchse der Bus-Platine; Übertragungsrate 1200 Baud).

RI (Festadresse #0046)
Liest ein Byte vom schnellen Cassetten-Interface ins Register A (Akkumulator) ein (Anschluß an der TB-Buchse der Bus-Platine; Übertragungsrate 1200 Baud).

LO (Festadresse #004F)
Gibt das im Register C stehende ASCII-Zeichen an den externen Drucker aus (serielle Schnittstelle auf der Bus-Platine; Übertragungsrate 4800 Baud).

Monitor-Sprungverteiler

Beginnend bei JMPTAB (Adresse siehe Symboltabelle) sind die zu den Monitor-Anweisungen A...Z gehörenden Sprungbefehle angeordnet; das Sprungziel für A steht also in JMPTAB, das für B in JMPTAB + 3 usf. Am Ende dieser Tabelle steht ein 27. Sprungbefehl nach 8000, der über das Monitor-Kommando „Ä, Return" erreicht wird; dabei gelangen die hinter Ä eingegebenen Parameter in die RAM-Zellen SRCBEG, SRCEND und DSTBEG (Adressen siehe Symboltabelle) und können von dort abgerufen werden (gedacht als Hilfsfunktion, z. B. für Testzwecke).

Initialisierung des Video-Controllers

Bei dem im Video-Interface verwendeten Steuerbaustein lassen sich sämtliche Randbedingungen per Programm vorgeben („programmieren"); dazu gehören z. B. Zeilenanzahl, Zeichen pro Zeile, Plazierung und Dauer des vertikalen und horizontalen Synchronimpulses, um nur einige Parameter zu nennen. Diese Vorgaben, die für das große und kleine Bildformat natürlich unterschiedlich aussehen, sind in zwei Tabellen TABIG (Großformat; Adresse siehe Symboltabelle) bzw. TABSML (Kleinformat) enthalten, von wo sie ausgelesen und in den Steuerbaustein überschrieben werden (**Tabelle 9**).

Programmbeispiel Ergebnisanzeige

Um Ihnen einen Eindruck davon zu geben, was man mit den genannten Unterprogrammen alles anfangen kann, lassen Sie uns ein Beispiel erarbeiten, bei dem man Operationen auf Maschinenebene recht anschaulich auf dem Bildschirm verfolgen kann. Dazu greifen wir auch auf Routinen zurück, die zusammen mit dem roten Monitor vorgestellt worden sind (vgl. „Der MOPPEL-Profi-Monitor unter der Lupe"). Natürlich können Sie die folgenden Programmbeschreibungen nicht lesen wie einen Liebesroman, sondern es ist schon erforderlich, daß Sie sich die Erläuterungen zu den einzelnen Programm-Bausteinen näher ansehen und hautnah verfolgen. Nur so lernen Sie die genauen Zusammenhänge und den richtigen Umgang mit den beschriebenen Unterprogrammen kennen.
Wir wollen ein Programm DEMO erstellen, das den Akkumulator-Inhalt modifiziert und ihn anschließend sowohl hexadezimal wie auch als Bitmuster auf dem Bildschirm darstellt. Nach Betätigen einer Taste erfolgt der Sprung zum Programmanfang, so daß auf diese Weise eine Endlos-

Tabelle 9.
Parameter zum Initialisieren der Register im Video-Steuerbaustein.

TABxxx	Reg	
	0	Gesamtanzahl (−1) der Zeichen einer Zeile (bestimmt die Hor-Sync-Frequenz)
+ 1	1	Anzahl der pro Zeile dargestellten Zeichen
+ 2	2	Position des horizontalen Synchron-Impulses (beeinflußt den linken und rechten Bildrand)
+ 3	3	Dauer des vertikalen (obere 4 Bits) und horizontalen Synchron-Impulses (untere 4 Bits)
+ 4	4	Gesamtanzahl (−1) der Textzeilen
+ 5	5	Anzahl der ergänzenden Bildzeilen (mit Reg 4 so einstellen, daß die Vert-Sync-Frequenz möglichst nahe der Netzfrequenz entspricht
+ 6	6	Anzahl der dargestellten Zeilen
+ 7	7	Position des vertikalen Synchron-Impulses
+ 8	8	Betriebsart (= 30 HEX)
+ 9	9	Bildzeilen (−1) pro Textzeile (einschließlich etwaiger Zwischenräume)
+ A	10	Cursor-Mode (aus bzw. langsam/schnell/gar nicht blinkend) sowie Bildzeilen-Nr. der Cursor-Oberkante
+ B	11	Bildzeilen-Nr. der Cursor-Unterkante
+ C	12	Video-RAM-Adresse (obere Hälfte), bei der die Darstellung auf dem Bildschirm beginnt
+ D	13	untere Adreßhälfte zu Reg 12
+ E	–	Anfangsadresse (untere Hälfte) der untersten Textzeile
+ F	–	Anfangsadresse (obere Hälfte)

Die Angaben beziehen sich auf den Steuerbaustein R6545-1 der Firma Rockwell; Äquivalenztypen anderer Hersteller weichen hiervon z. T. erheblich ab.

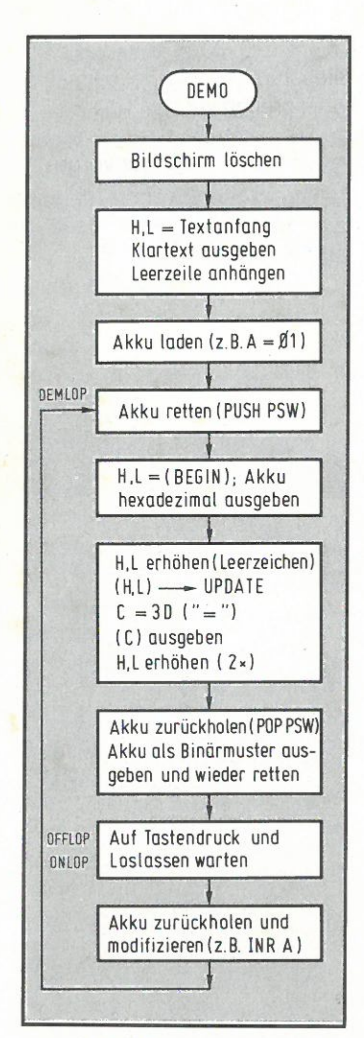


Bild 134: Trotz des simplen Programmbeispiels sind beim Bildschirmbetrieb umfangreiche Vorbereitungen zu treffen.

schleife entsteht. Vor dem Einsprung in diese Endlosschleife soll noch die Ausgabe eines Textes erfolgen, damit das Ganze einen etwas schöneren Anstrich erhält; wenn wir schon mit dem Bildschirm operieren, dann eben auch richtig (**Bild 134**).

Um einen Klartext zu generieren, gehen Sie wie folgt vor: Löschen Sie den Bildschirm (per J-Kommando) und schreiben Sie nach links oben den gewünschten Text (mit den Cursor-Steuerbefehl „Home" = CTL+Z kommen Sie in die linke obere Ecke). Anschließend retten Sie den Inhalt des Video-RAMs per R-Kommando nach 8000ff.

Achtung! Das „R" muß in der untersten Zeile hinter „Anweisung an MOPPEL >" eingegeben werden. Sie gelangen dorthin, indem Sie den Cursor erst nach links oben bringen, dann (per Überlauf!) nach unten schieben (CTL+U) und anschließend dreimal TAB drükken. Sie können sich übrigens sofort davon überzeugen, daß dieses Überschreiben geklappt hat: Das Kommando Q holt die nach 8000 geretteten Daten zurück auf den Bildschirm (ohne sie in 8000ff. zu löschen). Um das am Textende erforderliche Null-Byte einzugeben, schließen Sie die Texteingabe einfach mit CTL+Ä ab; der dabei generierte „Klammeraffe" (At-Zeichen) hat nämlich gerade den internen Code „00". Soviel zur Vorbereitung.

Um den Bildschirm im Anwenderprogramm zu löschen, rufen wir das Unterprogramm CLRVID auf; damit bekommen wir eine aufgeräumte, saubere Darstellung. Danach folgt die Ausgabe des zuvor nach 8000 überschriebenen Klartextes (das Null-Byte am Ende nicht vergessen, sonst gibt es Programmsalat!). Dazu ist H,L auf den Textanfang 8000 zu setzen und das Unterprogramm STRING aufzurufen (**Bild 135**). Interessanter wird das Ganze, wenn Sie statt STRING UNDRLN einsetzen; dann nämlich erscheint Ihre Texteingabe in Inversdarstellung (**Bild 136**). Um die nachfolgenden Ausgaben optisch ein bißchen zu trennen, hängen wir zweimal einen Zeilenvorschub an (Unterprogramm LINFED). Jetzt wird der Akkumulator mit demjenigen Anfangswert geladen, mit dem das ganze Spiel losgehen soll, also z. B. A=01. Mit dem anschließenden Sprungziel DEMLOP beginnt die Endlosschleife unseres Programms; hier wird erst einmal der Akku-Inhalt gerettet (per PUSH-Befehl in den Stack überschrieben), weil die nachfolgenden Operationen auf den Akku zugreifen und seinen Inhalt durcheinanderbringen. Die Darstellung des Akkumulator-Inhalts auf dem Bildschirm übernimmt das Unterprogramm BYTOT; vor dessen Aufruf muß das Registerpaar H,L mit derjenigen Adresse des Video-RAMs geladen werden, bei der die Ausgabe beginnen soll. Wenn dies links unten erfolgen soll, genügt es, die passende Anfangsadresse aus 3784/85 auszulesen (s. o.).

Um an die hexadezimale Datenausgabe auf dem Bildschirm ein Leerzeichen anzuhängen (das sieht gefälliger aus), wird das Zielregister H,L um Eins erhöht und nach 3782/83 transportiert (dort erwartet das gleich folgende Unterprogramm CO die Zieladresse für seine Zeichenausgabe). Um die Darstellung noch weiter zu schönen (und damit Sie die Handhabung von CO kennenlernen), soll zwischen hexadezimaler und binärer Darstellung ein „="eingefügt werden; der ASCII-Code dafür lautet 3D, und wenn der ins C-Register geladen wird, gefolgt vom Aufruf CALL CO, taucht auf dem Bildschirm das Gleichheitszeichen auf. Vor der folgenden Binärausgabe wird H,L um Zwei er-

```
2800                    ORG    2800H

2800 CD 2D 10   DEMO    CALL   CLRVID     VIDEO-RAM LÖSCHEN
                *
2803 21 00 80           LXI    H,8000H    ANFANG TEXTBUFFER
2806 CD 2A 10           CALL   UNDRLN     ZEICHENFOLGE AUSGEBEN
2809 CD A2 12           CALL   LINFED
280C CD A2 12           CALL   LINFED
                *
280F 3E 01              MVI    A,01       ANFANGSWERT LADEN
                *
2811 F5         DEMLOP  PUSH   PSW        AKKU RETTEN
                *
2812 2A 84 37           LHLD   3784H      ADRESSE LINKS UNTEN HOLEN
2815 CD 3F 10           CALL   BYTOT      UND AKKU HEXADEZIMAL AUSGEBEN
                *
2818 23                 INX    H          FÜR LEERZEICHEN
2819 22 82 37           SHLD   3782H      H,L NACH UPDATE
281C 0E 3D              MVI    C,3DH      ASCII-CODE FÜR "O="
281E CD 49 00           CALL   CO         BILDSCHIRMAUSGABE
2821 23                 INX    H          NEUE ZIELADRESSE
2822 23                 INX    H
                *
2823 F1                 POP    PSW        AKKU ZURÜCKHOLEN
2824 CD 42 10           CALL   BYTBIN     UND BINÄR AUSGEBEN
2827 F5                 PUSH   PSW        AKKU ERNEUT RETTEN
                *
2828 CD 52 00   OFFLOP  CALL   CSTS       ASCII-STATUS ERMITTELN
282B CA 28 28           JZ     OFFLOP     KEINE TASTE: JMP
282E CD 52 00   ONLOP   CALL   CSTS       ASCII-STATUS ERMITTELN
2831 C2 2E 28           JNZ    ONLOP      TASTE: JMP
                *
2834 F1                 POP    PSW        AKKU ZURÜCKHOLEN
2835 3C                 INR    A          UND MODIFIZIEREN
2836 C3 11 28           JMP    DEMLOP
```

Bild 135: Gut 50 Bytes belegt das Programm im Arbeitsspeicher (den Textbuffer nicht mit eingerechnet).

höht, um die passende Anfangsadresse für diese Ausgabe bereitzustellen.
Nachdem der Akkumulator durch die letzten Operationen arg durcheinandergewürfelt worden ist, laden wir ihn mit dem oben geretteten Anfangswert (POP-PSW-Befehl), geben ihn binär aus (Unterprogramm BYTBIN) und schreiben ihn gleich wieder in den Stack.
Um die Endlosschleife nur auf Tastendruck fortzusetzen, sind die beiden Unterschleifen OFFLOP und ONLOP eingebaut; in beiden erfolgt die Abfrage, ob eine (beliebige) Taste gedrückt ist (Unterprogramm CSTS), nur der Schleifenaussprung hängt von unterschiedlichen Bedingungen ab: OFFLOP wird so lange durchlaufen, wie das Zero-Flag nach dem CSTS-Rücksprung auf HIGH ist (springe nach OFFLOP, wenn Z=1, d. h. keine Taste gedrückt ist); hier wartet das Programm darauf, daß eine Taste angeschlagen wird.
In ONLOP passiert das Umgekehrte, nämlich der Schleifendurchlauf, bis das *Loslassen* der Taste erkannt worden ist (das Zero-Flag nicht mehr LOW ist). Diese Sequenz stellt sicher, daß pro Tastendruck wirklich nur ein Schleifendurchlauf von DEMLOP erfolgt; andernfalls würde sich bei gedrückter Taste ein rasender Durchlauf an den anderen reihen, so schnell es der Prozessor eben schafft.
Der Rest ist schnell abgehandelt, obwohl doch erst hier die eigentliche Manipulation erfolgt (in Adresse 2835); dazu holen wir uns den zuvor geretteten Akku-Inhalt aus dem Stack zurück, und dann modifizieren wir ihn (beispielsweise durch Hochzählen, denkbar ist natürlich auch ein Schiebebefehl oder alles andere). Der Sprung nach DEMLOP schließt die Schleife, und wenn Sie das Programm eingegeben und gestartet haben, sehen Sie (hexadezimal und binär), wie sich bei jedem Tastendruck der Akku-Inhalt um ein Bit erhöht.
Übrigens: Mit einer winzigen Änderung erreichen Sie bei diesem Programm eine ganz interessante Mehrleistung; Sie brauchen nur den bedingten Sprungbefehl in 282Bff. durch den Unterprogramm-Aufruf CALL ONSEC (Adresse siehe Symboltabelle) zu ersetzen, dann geht der Schleifendurchlauf vollautomatisch im Sekundentakt vor sich. Wenn Sie bei einem besonders interessanten Zählerstand verharren wollen, genügt das Niederhalten einer Taste; erst nach dem Loslassen (und dem Verlassen von ONLOP) geht es dann weiter.

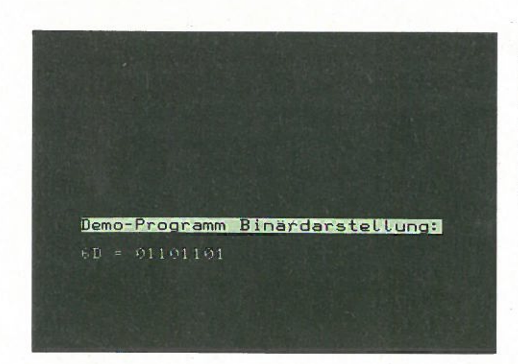
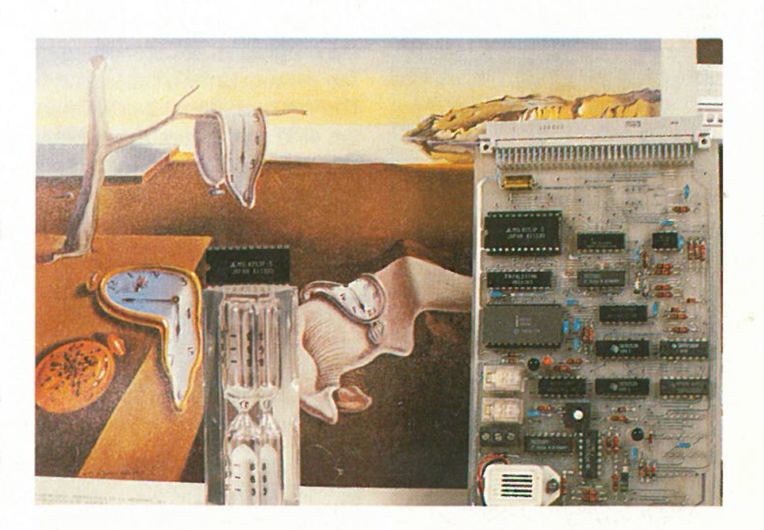


Bild 136: Mit den beschriebenen Programmfunktionen ergibt sich eine anschauliche Bildschirmdarstellung.

Wenn dieses Beispielprogramm wie beschrieben läuft, probieren Sie in 2835 auch mal einen anderen Befehl aus, z. B. das Linksschieben oder die Addition eines anderen Registerinhalts. Sie glauben nicht, wie man sich bereits an diesen kleinen Dingen des (Programmierer-)Lebens erfreuen kann, auf die Sie mit Recht stolz sein können, weil Sie ja einiges dazu getan haben; und interessanter als jede amerikanische Fernsehserie ist das allemal!

Wohldosierte Zeitpartikel



Der Einsatz eines Timer-ICs reicht vom programmierbaren Rechteckgenerator über einstellbare Verzögerungszeiten bis hin zum Frequenzzähler; bei Bedarf liefert Ihnen das unscheinbare IC Signale, die sich nur alle paar Jahre wiederholen!

Ein Zählerbergwerk

In sehr vielen Anwendungsfällen ist ein Mikrocomputer mit der sinnlosen Tätigkeit beschäftigt, seine Zeit zu verschwenden, d. h. auf das eine oder andere Ereignis eine definierte Zeit lang zu warten. Dabei gibt es eine Vielzahl unter-

Bild 137: Drei separate 16-Bit-Rückwärtszähler im 8253 lassen sich für die unterschiedlichsten Anwendungsfälle individuell programmieren.

Tabelle 10. Betriebsarten des Timer-ICs 8253

Mode	Beschreibung
Mode 0:	Impulserzeugung (einmalig, Software-Auslösung) LOW-Impuls am Ausgang mit n Taktperioden Dauer Beginn: Einschreiben des Anfangswertes n
Mode 1:	Impulserzeugung (einmalig, Hardware-Auslösung) LOW-Impuls am Ausgang mit n Taktperioden Dauer Beginn: Positive Flanke am Gate-Eingang
Mode 2:	Teiler durch n LOW-Impuls am Ausgang mit der Dauer einer Taktperiode, zyklisch alle n Takte wiederkehrend
Mode 3:	Rechteckgenerator Rechtecksignal am Ausgang mit der Periodendauer von n Taktperioden; bei ungeradem n ist der HIGH-Zustand eine Taktperiode länger als der LOW-Zustand
Mode 4:	Trigger-Impuls (einmalig, Software-Auslösung) LOW-Impuls am Ausgang mit der Dauer einer Taktperiode, beginnend nach n Taktperioden Zählbeginn: Einschreiben des Anfangswertes n
Mode 5:	Trigger-Impuls (einmalig, Hardware-Auslösung) LOW-Impuls am Ausgang mit der Dauer einer Taktperiode, beginnend nach n Taktperioden Zählbeginn: Positive Flanke am Gate-Eingang

n: Anfangswert, der in den Zähler geladen wird
Taktperiode: Bezieht sich auf das CLOCK-Signal

stützender Bausteine, die einer vielbeschäftigten Zentraleinheit solche Nebentätigkeiten abnehmen (Zeitgeber- oder Timer-ICs), damit die CPU die Hände frei behält für „echte" Aufgaben. Ein typischer Vertreter programmierbarer Zeitgeber-Schaltungen ist der 8253 von Intel. Dieses IC enthält drei voneinander unabhängige 16-Bit-Rückwärtszähler, deren ausgangsseitiges Verhalten vielseitig programmierbar ist; jeder Zähler besitzt einen eigenen Takteingang (CLOCK), der mit dem zugehörigen Freigabe-Eingang (GATE) UND-verknüpft ist, d. h. die Taktimpulse am Eingang bewirken nur dann ein Weiterzählen, wenn das Gate auf HIGH-Pegel liegt (**Bild 137**).
Das Zählverhalten (binär oder BCD) sowie das Verhalten des Ausgangs (Betriebsart oder zu neudeutsch *Mode*, gesprochen „Mohd") lassen sich für jeden Zähler separat vorgeben; dazu lädt der Anwender ein sogenanntes Steuerwort ins Control-Register des ICs (für jeden 16-Bit-Zähler wieder ein eigenes), und fortan weiß der Baustein, wie er seine Zähler-Schäfchen zu behandeln hat. Die sechs möglichen Betriebsarten dieses ICs stellt **Tabelle 10** zusammen. Natürlich lassen sich auch die Anfangswerte getrennt laden, von denen aus die Zähler ihr Abwärtszählen starten sollen. Dazu kann man nacheinander zwei Bytes in die 16 Zählerbits überschreiben, wenn eine so feine Auflösung gefordert ist; man kann aber zur Vereinfachung auch nur die untere oder nur die obere Hälfte der Zähler laden (die andere Hälfte ist dann automatisch Null), wenn das der Anwendungsfall zuläßt. Schließlich besteht auch noch die Möglichkeit, den Zählerstand während des Betriebs zu lesen, damit man ihn bei Bedarf weiterverarbeiten kann. Alles in allem steckt in so einem niedlichen IC so viel an Leistung, daß man seine Mühe hätte, all das in „diskreter" TTL-Technik auf einer einzigen Europa-Karte unterzubringen!

Kein Beinsalat

Sehen wir uns an dem „Maikäfer" einmal die 24 Beine und deren Bedeutung an. Glauben Sie nur nicht, daß alle 48 Q-Ausgänge der drei Zähler einzeln herausgeführt sind, denn dann wäre der 8253 ein Kuchenblech-IC, an dem niemand mehr seine Freude hätte! Statt dessen besitzt der Baustein acht bidirektionale Anschlüsse, die mit dem System-Datenbus verbunden werden; drei zusätzliche Steuerleitungen CS (*Chip Select*), RD (*Read*) und WR (*Write*) übernehmen die Verwaltungsarbeit: Egal, was um ihn herum passiert, spitzt das IC nur dann seine Ohren, wenn CS auf LOW ist. Dies auszulösen, ist Aufgabe der Selektierungslogik, die unabdingbarer Bestandteil jeder Mikrocomputer-Hardware ist. Nach der Aktivierung gibt das WR- *oder* das RD-Signal die Übertragungsrichtung des Datenflusses an: Bei WR=LOW soll (von der CPU) in den Baustein geschrieben werden, beispielsweise zum Laden eines Steuerwortes oder eines Zähler-Anfangswerts. Bei LOW an RD sollen Daten aus dem Baustein (in die CPU) gelesen werden, etwa der augenblickliche Zählerstand. Und schließlich muß dem Timer noch klargemacht werden, für welchen Zähler ein Steuerwort oder ein Anfangswert bestimmt ist; dazu dienen die beiden Eingänge A0 und A1, die an zwei Adreßbits im System angeschlossen sind, und die je nach Zustand eins von vier möglichen Zielen anwählen: Einen der Zähler 0, 1 oder 2 oder aber das Control-Register; letzteres besitzt nach außen hin übrigens nur eine Ansprech-Adresse, und zwei Bits innerhalb des Steuerwortes geben an, für welchen Zähler die Verwaltungsdaten gelten.
Betrachten wir Aufbau und Bedeutung des Steuerwortes einmal im Detail (**Bild 138**). Die obersten beiden Bits bestimmen, für welchen Zähler 0, 1 oder 2 das Steuerwort gilt; die Zähler-Nummer wird einfach mit zwei Bits binär codiert angegeben (00, 01 oder 10). Natürlich können Sie so ein armes IC damit ärgern, als Zähler-Nummer „11" anzugeben; denn in diesem Fall weiß der Timer beim besten Willen nicht, was er tun soll, und daher ignoriert er solche Eingaben (ähnliches gilt für die Spezifikation des Modes, wo von acht möglichen Zuständen nur sechs zulässig sind).

```
7  6  5  4  3  2  1  Ø
C  C  RW RW M  M  M  BC

                     Ø-binär
                     1-BCD
            Ø  Ø  Ø — Mode Ø
            Ø  Ø  1 — Mode 1
            X  1  Ø — Mode 2
            X  1  1 — Mode 3
            1  Ø  Ø — Mode 4
            1  Ø  1 — Mode 5
      Ø  Ø ——— Zählerstand Lesen
      Ø  1 ——— nur unteres Byte
      1  0 ——— nur oberes Byte
      1  1 ——— erst unteres,dann
               oberes Byte
Ø  Ø ——— Zähler Ø
Ø  1 ——— Zähler 1
1  Ø ——— Zähler 2
1  1 ——— unzulässig
```

Bild 138: Das Steuerwort im Control-Register bestimmt das Verhalten der Zähler; die obersten beiden Bits geben die Zähler-Nummer an.

Eine kleine Ewigkeit

Die nächsten beiden RW-Bits beziehen sich auf das Lesen des Zählerstandes („00") bzw. auf das Laden mit einem Anfangswert: bei 01 wird nur das untere Byte des 16-Bit-Zählers geladen, bei 10 nur das obere; das jeweils andere wird automatisch nullgesetzt. Bei

11 in diesen Bits erwartet der Timer nacheinander zwei Bits, von denen das erste in die untere und das zweite in die obere Zählerhälfte gelangen. Die drei Mode-Bits beziehen sich auf die in Tabelle 10 genannten Betriebsarten, und das niedrigstwertige Bit schließlich gibt an, ob die ganze Zählerei binär oder im BCD-Format ablaufen soll.
Beispiel: Zähler 0 soll durch 2048 teilen (z. B. als Rechteckgenerator im Mode 3); dann könnte er das im BCD-Mode tun, wenn seine obere Hälfte mit „20" und seine untere Hälfte mit „48" geladen werden. Die andere Möglichkeit besteht darin, Nr. 0 binär zählen zu lassen und nur seine obere Hälfte mit HEX „08" zu laden; zusammen mit der unteren Hälfte wird damit insgesamt ein Zählerstand 0800HEX vorgegeben, bei dem nur das Bit 11 (mit der Wertigkeit 2^{11}=2048) auf HIGH ist. Das zum zweiten Fall passende Steuerwort sieht wie folgt aus (vom höchstwertigen Bit Nr. 7 beginnend, nach rechts fortsetzend): 00 für „Zähler Nr. 0" gefolgt von 10 für „nur ein Byte in die obere Hälfte laden" (dieses Laden passiert später); dann kommen drei Bits mit 011 für „Mode Nr. 3", und im kleinsten Bit Nr. 0 steht eine „0" für binäres Zählverhalten. Zusammengesetzt ergibt sich die Bit-Struktur „00100110" für das Steuerwort (=26HEX), um Zähler 0 im gewünschten Mode zu betreiben. Erst nachdem dieses Steuerwort im Timer-IC abgelegt worden ist, kann die Vorgabe für den Anfangswert des Zählers dorthin überschrieben werden.
Damit Sie einmal ein Gefühl dafür bekommen, was drei so unscheinbare Zählerchen vermögen, machen Sie folgendes Gedankenexperiment mit (wenn Sie wollen, setzen Sie es auch in die Tat um): Angenommen, man schaltet hardwaremäßig alle drei 16-Bit-Zähler hintereinander (man „kaskadiert" sie), indem man den Ausgang des einen mit dem Eingang des nächsten verbindet. Eingangsseitig speist man einen Takt von 2 MHz ein (maximal zulässiger Wert) und programmiert die so entstandene 48-Bit-Zählerkette auf Mode 2 mit maximalem n, was meinen Sie, was in diesem Fall am Ausgang des letzten Zählers passiert? Nun das ist ganz einfach; es tritt ein LOW-Impuls von 0,5 µs Dauer auf (eine 2-MHz-Taktperiode lang) und zwar alle viereinhalb Jahre wiederkehrend! Wenn Sie es nicht glauben, teilen Sie 2 MHz einmal durch 2^{48}, dann wissen Sie auch, warum man diesen Versuch getrost nur gedanklich durchführt!
Welche Zieladressen Sie für Steuerwort und Anfangswert angeben müssen, das hängt ausschließlich von der Hardware-Struktur (Dekodier-Logik) des Systems ab; im Beitrag „Bits im Gänsemarsch" (Seite 70) finden Sie die Struktur für eine umfangreiche Port-Adreß-Dekodierung. Damit steht dann Ihrem programmierbaren Zeit-Impulsverzögerungs-Rechteckgenerator nichts mehr im Wege!

Einzeln ein- und aussteigen

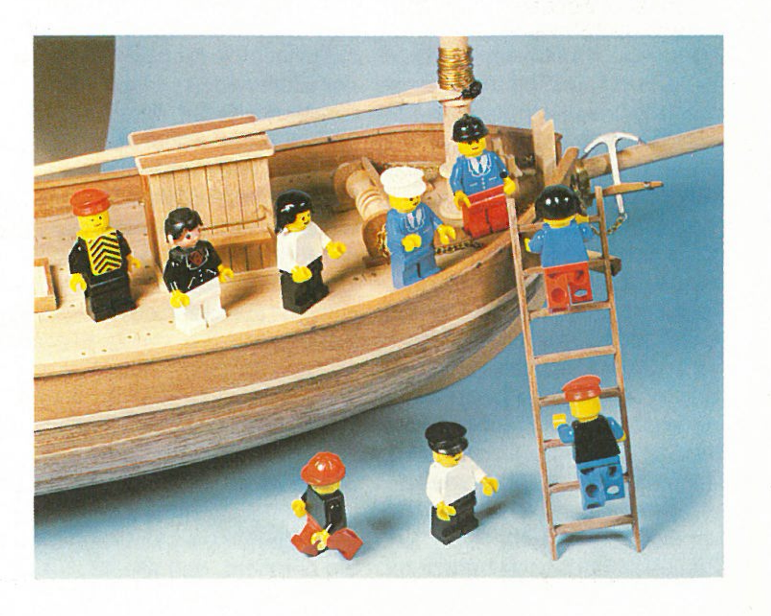
Die Ansteuerung von Peripheriegeräten (z. B. Drucker) erfolgt aus Gründen der Störsicherheit meist bitseriell. Zur Abwicklung dieses Datenverkehrs setzt man spezielle Interface-Bausteine ein, die die Zentraleinheit entlasten.

Hochtrabende Namen

Die für den seriellen Datenverkehr angebotenen Bausteine tragen meist sehr anspruchsvolle Namen, die Schuld sind an dem übertriebenen Respekt, den viele Computer-Fans derartigen ICs entgegenbringen: „Peripherer Interface-Adapter" (PIA) ist da noch harmlos gegen „Universeller asynchroner/synchroner Empfänger/Sender" (UART) oder gar „Asynchroner Kommunikations-Interface-Adapter" (ACIA), wie man den hier betrachteten Interface-Baustein 6850 (von Motorola) getauft hat; wir haben ihn nicht etwa wegen seines Namens, sondern seiner besonderen Leistungsfähigkeit ausgewählt und wollen uns ansehen, wie er zu behandeln ist, um den Datentransfer wie gewünscht abzuwickeln.
Im Prinzip ist so ein ACIA auf der Sendeseite ein Parallel-/Serien-Wandler und auf der Empfangsseite ein Serien-/Parallel-Wandler, d. h. er zerstückelt die vom Mikrocomputer gelieferten parallelen Daten (8-Bit-Worte) und gibt sie klekkerweise aus. Beim Entgegennehmen der ebenfalls bitweise ankommenden Daten setzt er sie umgekehrt wieder zusammen und übergibt sie als 8-Bit-Datenwort an die CPU.
Diesen Umstand der seriellen Datenübertragung macht man aus zwei Gründen: Erstens kommt man dabei mit einer einzigen Datenleitung aus (gegenüber acht bei der parallelen Übertragung) und zweitens erreicht man dabei (in Verbindung mit höheren Spannungspegeln) eine bessere Störsicherheit, wobei die erforderlichen Treiberschaltungen beim seriellen Verkehr eben auch nur einmal vorhanden sein müssen.

Invasion von Bits

Wenn auf jemanden ein Strom aus seriellen HIGH- und LOW-Bits einstürmt, ist es nützlich zu wissen, wann ein altes Datenwort zu Ende ist und ein neues anfängt; andernfalls gibt es Bit-Salat, bei dem nichts sinnvolles herauskommt. Daher erweitert man ein Datenwort bei der seriellen Übertragung „vorne" um ein Startbit (immer LOW) und hängt „hinten" ein Stopbit an, das immer HIGH ist (**Bild 139**). Dazwischen rasseln die Datenbits heraus, beginnend beim LSB, also genau umgekehrt zur gewohnten Darstellungsweise, wo das niedrigstwertige Bit rechts steht. Zur Synchronisation muß die empfangende Stelle nur den ersten LOW-Impuls abwarten, sammelt die darauffolgenden acht Datenbits ein und weiß, daß sich daran ein HIGH-Stopbit anschließt. Durch den so ent-



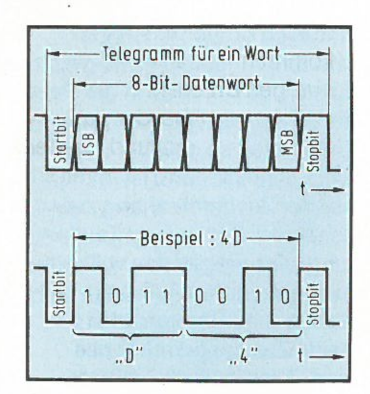


Bild 139: Zum Zweck der Synchronisation beim Empfang erweitert man ein Datenwort bei der seriellen Übertragung um je ein Start- und Stopbit.

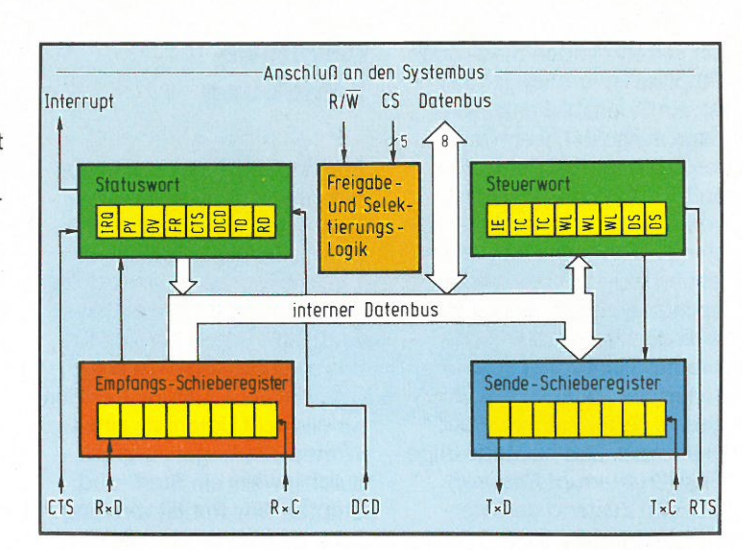
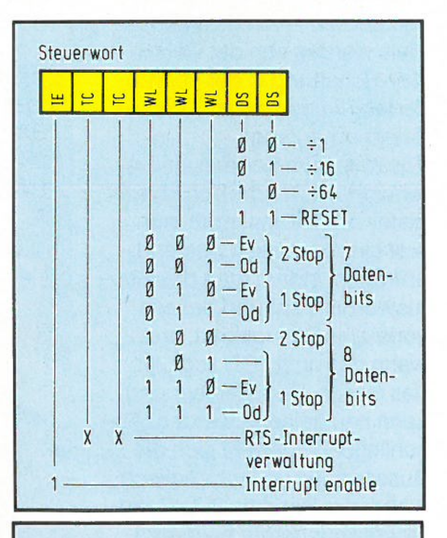


Bild 140: Für das Senden und Empfangen der Daten besitzt der Interface-Baustein 6850 zwei vollkommen getrennte Zweige.

stehenden Rahmen ist das Einsammeln und Zusammensetzen der einzelnen Bits kein Problem mehr, zumal der Interface-Baustein all das eigenständig übernimmt, nachdem ihm der Programmierer die Randbedingungen vorgegeben hat. Streng genommen könnte man ohne jedes Stopbit auskommen, weil die empfangende Stelle ja von der ersten eintreffenden negativen Flanke aus weiterzählen könnte, um jedes folgende Bit in dessen Mitte abzutasten; da aber bei kilometerlangen Bitkolonnen diese Synchronisation verloren gehen kann, schafft das zusätzlich eingefügte Stopbit nach jedem Wort wieder klare Verhältnisse.

Beim Anblick des ACIA-Innenlebens erkennen Sie schnell, daß dieses IC zu Recht den Zusatz „intelligent" trägt (**Bild 140**); die Leistungsfähigkeit geht nämlich doch schon ein gutes Stück über die reine Serien-/Parallel- bzw. Parallel-/Serien-Wandlung hinaus. Wie üblich, wird auch dieses Interface-IC über den Datenbus mit der Zentraleinheit verbunden. Insgesamt fünf Eingänge werden dabei an die System-Auswahllogik angeschlossen, um entsprechend der gewählten Adreßzuordnung den Baustein nur dann zu aktivieren, wenn die CPU die ihm zugewiesene Adresse ausgibt; die Schreib-/Lese-Leitung R/W gibt zusätzlich die Übertragungsrichtung des Datenflusses an: Ist sie HIGH, werden Daten aus dem IC gelesen (aus dem Empfangs-Schieberegister oder dem Statusregister), ist sie LOW, überschreibt die CPU Daten in den Interface-Baustein (ins Sende-Schieberegister oder ins Control-Register).

Das Steuerwort im Control-Register bestimmt die einzelnen Randbedingungen der Übertragung (**Bild 141**). Die beiden untersten Bits „DS" (*Divide Select*) geben an, ob der an T×D bzw. R×D anliegende Schiebetakt ungeteilt oder durch 16 bzw. 64 geteilt werden soll, ehe er das zugehörige Schieberegister taktet. Die drei Bits „WL" für die Wahl der Wortlänge erlauben eine vielseitige Gestaltung des Telegramms; es ist die Übertragung von wahlweise 7 oder 8 Datenbits möglich, gefolgt von entweder einem oder zwei Stopbits. Außerdem kann beim Senden ein zusätzliches Paritätsbit hinzugefügt werden, das beim Empfangen automatisch geprüft wird (wahlweise gerade oder ungerade Parität). Dahinter verbirgt sich folgende Leistung: Bei jedem zu übertragenen Wort zählt der Baustein selbsttätig die Anzahl der darin enthaltenen HIGH-Bits; mit dem Paritätsbit ergänzt er diese Anzahl auf einen geraden (*Parity Even*) oder ungeraden (*Parity Odd*) Wert, so daß man auf diese Weise eine erste, grobe Prüfung auf Übertragungsfehler erhält. Die obersten drei Bits im Steuerwort dienen zur Verwaltung des Interrupt-Ausgangs; der kann mit der CPU verbunden werden, um beispielsweise immer dann ein Unterbrechungs-Signal zu erzeugen, wenn das Empfangs-Schieberegister voll ist, damit dann (und auch wirklich nur dann) das nächste zu übertragende Wort geladen wird. In der Zwischenzeit kann sich die Zentraleinheit mit anderen Dingen befassen, denn sie weiß ja, daß sie der Interface-Baustein wieder anstößt, wenn dies erforderlich ist.



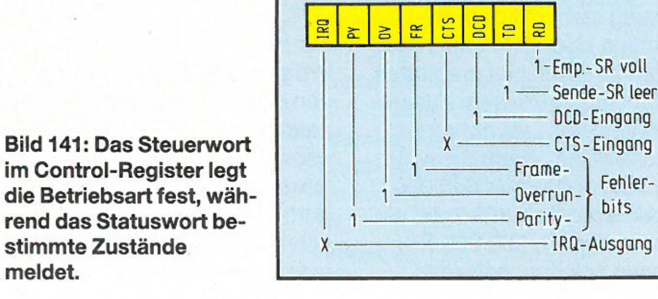


Bild 141: Das Steuerwort im Control-Register legt die Betriebsart fest, während das Statuswort bestimmte Zustände meldet.

Zustand zufriedenstellend

Das Statuswort ermöglicht es, den Zustand verschiedener Anschlußleitungen bzw. Register abzufragen. Die untersten beiden Bits melden (bei HIGH-Pegel), daß das Empfangs-Schieberegister voll bzw. das Sende-Schieberegister leer ist; in beiden Fällen muß die CPU reagieren, indem sie die Daten aus dem Empfänger herausholt oder neue Daten in den Sender nachschiebt. Die beiden nächsten Bits „DCD" und „CTS" melden schlicht und einfach den Zustand der entsprechenden Eingänge DCD und CTS (s. u.). Die drei Bits „FR", „OV" und „PY" dienen zur Fehleranzeige. Ein Rahmen-(Frame-)Fehler liegt vor, wenn beispielsweise eines

der einrahmenden Start- oder Stopbits verlorengegangen ist; ein Überlauf-Fehler wird dann ausgelöst, wenn im Datenstrom ein Zeichenverlust auftritt, etwa dann, wenn ein volles Empfangs-Schieberegister nicht ausgelesen worden ist und das nächste Zeichen bereits „einläuft"; Paritätsfehler liegen immer dann vor, wenn bei der Paritätsprüfung im Empfänger ein anderes Ergebnis auftritt als zuvor programmiert. Das höchstwertige Bit IRQ (*Interrupt Request*) gibt den Zustand der Interrupt-Leitung an, so daß dieser Pegel auch softwaremäßig abgefragt werden kann.
An die „Außenwelt" (d. h. zu den peripheren Geräten) führen insgesamt sieben Leitungen: Ausgangsseitig die Sende-Datenleitung T×D (*Transmit Data*) und der Sende-Takt T×C (*Transmit Clock*) und eingangsseitig die Empfangs-Datenleitung R×D (*Receive Data*) und der Empfangs-Takt R×C (*Receive Clock*). Am Anschluß RTS (*Request to Send*) gibt das IC an seine angeschlossene Außenstelle die dezente Aufforderung ab, Daten 'rüberzuschicken, und am korrespondierenden Eingang CTS (*Clear to Send*) vernimmt der ACIA, daß beim Peripheriegerät alles klar ist für den Datentransfer. Das DCD-Bein (*Data Carrier Detect*) ermöglicht schließlich die automatische Überwachung der angeschlossenen Außenstelle; bei Pegeländerungen an diesem Anschluß (d. h. bei Verlust der Rückmeldung) kann ein Unterbrechungssignal ausgelöst werden, um beispielsweise eine Fehlermeldung auszugeben.
Die gebotenen Leistungen der Paritätsprüfung oder automatischen Interrupt-Erzeugung muß man natürlich nicht wahrnehmen, wenn man darauf verzichten kann oder will. Aus Gründen der Übersichtlichkeit wollen wir uns bei der Betrachtung der prinzipiellen Funktionsweise auch nur auf das Wesentliche beschränken; ausführliche Hinweise für den Einsatz dieses Bausteins finden Sie im Beitrag „Bits im Gänsemarsch" ab Seite 70.

Geplante Schiebung

Beginnen wir beim Laden des Steuerwortes, das die ACIA-Betriebsart festlegt. Das an den Takteingängen R×C bzw. T×C anliegende Signal soll ungeteilt als Schiebetakt wirken, so daß die untersten beiden Bits auf LOW liegen müssen (vgl. Bild 140). Natürlich wollen wir alle acht Datenbits eines Wortes übertragen, ergänzt durch jeweils ein Start- und Stopbit; eine Paritätsprüfung schenken wir uns im Augenblick, so daß die Wortlänge mit dem Bitmuster „101" festgelegt wird. Die Übertragung von nur sieben Datenbits ist übrigens dafür vorgesehen, daß man es ausschließlich mit ASCII-Zeichen zu tun hat, die eben maximal nur sieben Bits lang sind. – Wenn uns im Moment auch die Interrupt-Verwaltung nicht weiter interessiert, lassen wir die obersten drei Bits im Steuerwort ebenfalls auf LOW, so daß sich damit ein Bitmuster „00010100" ergibt (=14HEX), das vor Beginn der Übertragung (einmalig) ins Control-Register zu laden ist. Die Hardware-Adressen der vier ACIA-internen Ziele werden von der Dekodier-Schaltung in der eben zitierten „Gänsemarsch"-Bauanleitung erzeugt.
Ehe man zum Senden ein Datenwort ins Sende-Schieberegister laden kann, muß man erst einmal dessen Zustand abfragen, indem man das Statuswort einliest und dessen vorletztes Bit maskiert; erst wenn das auf HIGH liegt, ist das Sende-Register leer und kann neu geladen werden. Anschließend vollzieht sich die Aussendung vollautomatisch, wobei der Sendetakt T×C die Übertragungsrate bestimmt. Da dieser Takt in keinerlei Beziehung steht mit dem Systemtakt oder dem Laden des zugehörigen Schieberegisters, spricht man in diesem Zusammenhang von „asynchroner" Arbeitsweise. Wenn Sie wollen, können Sie hier 500 000 Impulse pro Sekunde oder einen Impuls alle 500 000 Sekunden anlegen; entsprechend schnell oder langsam kleckern eben Ihre Datenbits über die Leitung!
Ähnlich automatisch (und asynchron) vollzieht sich das Empfangen. Mit dem Empfangstakt R×C gelangen die Datenbits ins Empfangs-Schieberegister, und wenn das gefüllt ist, geht das niedrigste Bit im Statuswort auf HIGH (bei entsprechender Vorbereitung kann dann auch ein Interrupt ausgelöst werden). Wenn die CPU ständig das Statuswort einliest, dessen unterstes Bit maskiert und auf HIGH-Pegel abfragt, erkennt sie daraus, wann ein vollständiges Datenwort zum Abholen bereitliegt. Bei entsprechender Programmierung des Steuerwortes übernimmt der Baustein auch noch vollautomatisch die Paritätsprüfung, so daß damit schon eine gewisse Gewähr für fehlerfreie Übertragung besteht.
Natürlich brauchen Sende- und Empfangstakt nicht gleich zu sein, obwohl das in der Praxis schon aus Gründen der Einfachheit so gewählt werden wird; entscheidend ist immer nur der Zusammenhang zwischen der Daten- und Taktleitung, was wegen des vollkommen getrennten Aufbaus von Sende- und Empfangsteil für beide Zweige getrennt passiert. Alles in allem bietet so ein ACIA doch schon einen ganz ansprechenden Komfort, der die Zentraleinheit beim Datentransfer weitgehend entlastet (darum sehen wir ihm seinen hochgestochenen Namen auch gern nach). Eine komplette V.24- oder Stromtreiber-Schnittstelle wird daraus aber erst, wenn die entsprechenden Treiber-Schaltungen noch ergänzt werden (±12-V-Logik bzw. Konstantstromquelle; vgl. Bauanleitung).

Daten vom laufenden Band



Zur externen Speicherung von Programmen und Daten eignen sich herkömmliche Kompakt-Kassetten; wählt man dabei als Aufzeichnungsverfahren die Phasen-Kodierung, geht dies nicht nur besonders schnell, sondern auch noch extrem störsicher.

Bandsalat voll Bits

Bei der Datenübertragung zwischen Mikrocomputer und peripheren Geräten hat man (zumindest theoretisch) die Möglichkeit, ein 8-Bit-Datenwort entweder über acht *parallele*

Leitungen oder *seriell* ein Bit nach dem anderen zu übertragen. Die Qual der Wahl, für welches der beiden Verfahren wir uns beim Magnetbandgerät entscheiden, findet eine rasche Auflösung: Da die wenigsten Mikrocomputer-Enthusiasten über eine 8-Kanal-Bandmaschine verfügen, entscheiden wir uns für die bitserielle Aufzeichnung und Wiedergabe; denn *eine* Aufzeichnungsspur haben sie alle, die großen und kleinen Rekorder, und die Ansprüche an Gleichlaufkonstanz oder Frequenzgang sind so gering, daß man als Bandgerät nahezu alles hernehmen kann, was auf Grabbeltischen im Kaufhaus oder in Kellerschubladen daheim herumliegt. Eins aber sollte für Sie von vornherein klar sein, wenn Sie unter die Tonmeister der Mikrocomputerei gehen: Nur beste Bandsorten (HiFi/Low Noise) gewährleisten einen einwandfreien Betrieb, denn bereits der kleinste Fehler im Band kann zu einem Bitverlust führen, der das beste Programm zum Krüppel macht!
Die Magnetbandaufzeichnung von Computer-Programmen hat sei jeher die (computerisierte) Menschheit fasziniert, weil dies ein extrem preiswertes Verfahren ist, sich eine ganze Bibliothek von Software zuzulegen. Probleme ergeben sich hierbei auf zwei Gebieten: Erstens soll das gewählte Aufzeichnungsverfahren nur geringe Ansprüche an den Rekorder stellen, damit man auch Billigstgeräte verwenden kann (s. o.); Hobbyisten sind in der Regel nicht gerade mit Geldmengen gesegnet. Und zweitens stellt sich immer wieder die Frage nach der Kompatibilität (= Verträglichkeit) zwischen verschiedenen Geräten; kann eine auf *einem* Computer aufgenommene Kassette von einem anderen Gerät gleichen Typs auch wieder einwandfrei gelesen werden?
Und so sind im Laufe der Zeit die unterschiedlichsten Aufzeichnungsverfahren entstanden und wieder verschwunden, alle mit mehr oder weniger gutem Erfolg: Man hatte zu Anfang damit begonnen, für HIGH-Bits in einem Datenwort einen kurzen Tonfrequenzimpuls aufzunehmen und für LOW-Bits eine Pause einzufügen; das sieht an sich ganz plausibel aus, hat aber bei den allseits vorhandenen Aufnahme-Automatiken zu lebhaften Protesten (und damit zu reichlichen Fehlern) geführt. Denn so eine Automatik weiß beim besten Willen nicht, worauf sie sich nun einregeln soll, wenn sie ab und zu knallharte Ton-Bits vorgesetzt bekommt, die andauernd durch ebenso harte Pausen unterbrochen werden.

Bits werden zu Musik

Die Folgerung lag auf der Hand: Man mußte nur für eine kontinuierliche Tonberieselung sorgen, indem man für die HIGH-Bits die eine und für LOW-Bits eine andere Frequenz erzeugt, beispielsweise gerade um eine Oktave (Verhältnis 2 : 1) auseinanderliegend (weil sich das elektrisch besonders einfach realisieren läßt, wie Sie als alter Flipflop-Freund längst erkannt haben). Nach diesem Verfahren der Frequenzumtastung hat sich dann tatsächlich eine gewisse Norm herausgebildet (Kansas-City-Standard; vgl. auch langsames Cassetten-Interface am Heftanfang), die nur einen Schönheitsfehler hatte: So recht kompatibel war auch dieses Verfahren nicht, und außerdem hatte das ganze Prinzip der Tonfrequenzaufzeichnung einen fundamentalen Nachteil: Angenommen, Sie wollen für ein Bit einen Tonfrequenzimpuls aufzeichnen, der später bei der Wiedergabe möglichst wieder erkennbar ist (weil sonst das Ganze wenig fruchtbar bleibt); wieviele Schwingungen spendieren Sie dann pro Bit, und welche Frequenzen sind (gerade bei billigen Bandgeräten) da noch vertretbar? Angenommen, man begnügt sich mit vier, fünf Perioden der gewählten Frequenz, deren unterster Wert (damit es schnell geht) bei ca. 1 kHz liegt (Periodendauer von 1 ms; die fünf Schwingungen pro Bit dauern also etwa 5 ms); bei einer Frequenzumtastung im Verhältnis 2 : 1 dauert die Übertragung eines einzigen Bytes dann im Mittel rund 35 ms (8 Datenbits, ergänzt durch Start- und Stopbit), und die Überspielung eines 1-K-Programms nudelt bereits über eine halbe Minute!

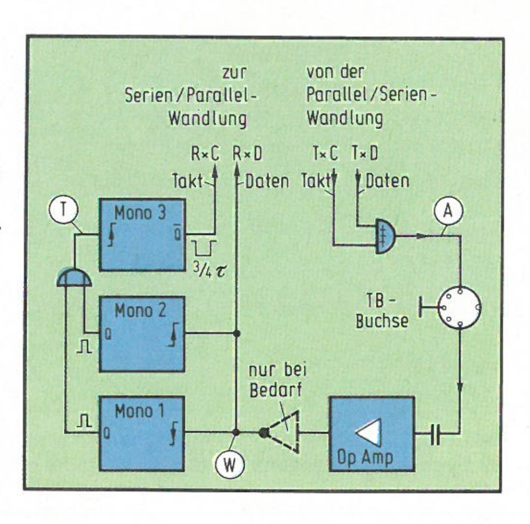


Bild 142: Blockschaltung für das phasenkodierte Aufzeichnungsverfahren, das trotz des geringen Aufwandes schnell und störsicher arbeitet.

Wegbereiter für eine bessere Zukunft waren hier diejenigen Taschenrechner früherer Tage, die Programme auf winzigen Magnetkärtchen abspeichern konnten. Für die nämlich galten hinsichtlich Gleichlaufschwankungen und Frequenzgang noch größere Toleranzen als bei den schlechtesten Kassetten-Rekordern, und das hat schlaue und hochbezahlte Leute auf das Verfahren der Phasen-Kodierung (*Phase Encoding*, abgek. PE) gebracht (das ein gutes Stück Genialität in sich birgt, wie Sie zugeben werden). Dabei erfolgt pro Informationsbit nur noch eine einzige Ummagnetisierung (gegenüber der vielfachen bei der Tonfrequenzaufzeichnung), was nach folgendem Schema abläuft: Das durch ein Start- und Stopbit erweiterte Datenwort wird nach alter Väter Sitte seriell ausgegeben, z. B. über einen asynchronen Interface-Baustein (vgl. Grundlagen-Beitrag „Einzeln aus- und einsteigen“ auf Seite 66); zusammen mit dem dabei verwendeten Schiebetakt gibt man dieses Signal auf ein Exklusiv-ODER-Gatter, das die Phasen-Kodierung vornimmt. Was dabei herauskommt, ist alles andere als aufregend (aber darum auch so genial!); sinnvoll wird das Ganze erst bei der Wiedergabe und Dekodierung (**Bild 142**).

Um sich selbst gedreht

Die vom Band kommenden Daten gelangen an einen Verstärker, der sie auf TTL-Pegel digitalisiert (und für die spätere, flankenabhängige Abtastung invertiert). Wenn im Bandgerät keine Vorzeichenumkehr stattfindet (bei einer geraden Anzahl von Verstärkerstufen), muß dies durch einen nachgeschalteten Inverter erreicht werden; Sie brauchen an dieser Stelle übrigens keine Transistoren im Bandgerät zu zählen, denn ob die eingelesenen Daten invers vorliegen oder nicht, das sieht sogar ein Hinterwäldler bei Mondfinsternis! In der Schaltung ist hier lediglich eine Brücke zu bestücken bzw. offen zu lassen. Am Ausgang des Inverters (bzw. des Operationsverstärkers) liegt dann ein Signal W vor, das zum phasenkodierten Verlauf A bei der Aufnahme genau invers verläuft. Daraus gilt es nun, die ursprünglichen Daten herauszulösen (**Bild 143**).
Zwei Monoflops Nr. 1 und Nr. 2 erzeugen bei jedem Pegelwechsel, also bei jeder positiven *und* negativen Flanke des

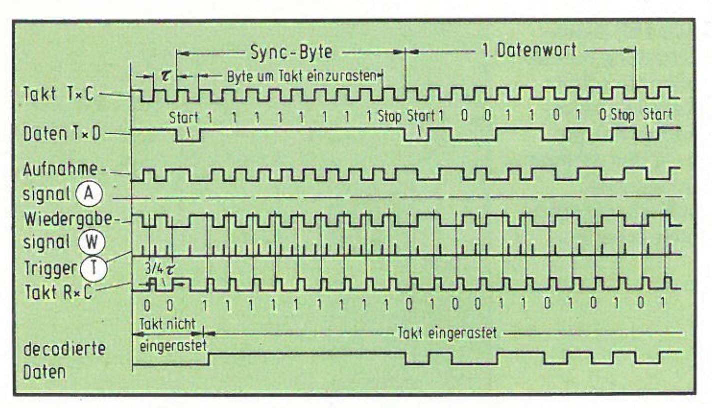


Bild 143: Aus den einzelnen Pegelverläufen wird der Einrastvorgang zu Beginn des Einlesens bei der Wiedergabe erkennbar.

digitalisierten Signals einen kurzen Pips (das ist ein kurzer Impuls, für den die Bezeichnung „Pieps" übertrieben wäre). Folgen zwei Informationsbits gleichen Pegels aufeinander, haben diese Impulse den Abstand einer Taktperiode; liegt zwischen zwei Informationsbits ein Pegelwechsel vor, dann beträgt der Abstand zwischen zwei Pipsen eine ganze Taktperiode (das kommt durch das aufnahmeseitige EXOR-Gatter, vgl. oben). Jeder Pips, egal, ob von der positiven oder negativen Flanke ausgelöst, triggert nun ein drittes Monoflop (Signal T), das ausgangsseitig einen LOW-Impuls genau definierter Dauer abgibt, nämlich von 75 % einer Taktperiode. Mit der positiven Flanke dieses Signals (also am Ende des 75-%-Impulses) wird der Pegel des Wiedergabe-Signals W abgetastet und übernommen; und dabei kommt genau das (durch Start- und Stopbit erweiterte) Datenwort wieder heraus! Natürlich ist es zweckmäßig, auch diese Aufgabe der Serien-/Parallel-Umsetzung wieder einem entsprechenden Interface-Baustein zu überlassen, der so etwas ohne viel Aufhebens erledigt; die entsprechenden Signalbezeichnungen (R×C, R×D für Empfangs-Takt und -Daten sowie T×C, T×D für Sende-Takt und -Daten) sind im Bild 142 daher auch eingetragen).

Pferdefüße im Galopp genommen

Wenn Sie sich den Impulsplan genau betrachten, erkennen Sie allerdings einen kleinen Pferdefuß: Erstens darf Monoflop Nr. 3 nicht retriggerbar sein, weil sonst kein Abtast-Takt entstehen würde; und zweitens darf die Triggerung von Mono 3 nicht wahllos von den Nr. 1- *oder* Nr. 2-Pipsen erfolgen, sondern wohlgeordnet vom jeweils richtigen (ansonsten würden die dekodierten Daten genau invers vorliegen, und damit ist schließlich niemandem gedient). Die richtige Zuordnung ergibt sich automatisch *beim ersten auftretenden Pegelwechsel*, bei dem sozusagen der Takt „einrastet". Damit dabei kein Datenwort verlorengeht, spendiert man bei der Aufnahme einfach ein blindes Byte vorweg, das einzig und allein dazu dient, den Takt bei der Dekodierung einrasten zu lassen (Sync-Byte). Und schon ist die Welt der Phasen-Kodierung wieder in Ordnung, und durch die niedrige Ummagnetisierungsfrequenz (einmal pro Bit) lassen sich hiermit problemlos Aufzeichnungsraten bis zu 4800 Bit/s erreichen; gegenüber den oben angestellten Überlegungen bedeutet dies eine Steigerung um den Faktor 10...15, ohne daß dies zu Lasten der Störsicherheit geht! Eine reale Schaltung für ein solches Kassetten-Interface finden Sie im Rahmen der Bauanleitung „Bits im Gänsemarsch" (s. u.).

Bits im Gänsemarsch

Beim Anschluß von Peripheriegeräten an einen Mikrocomputer spielt die serielle Datenübertragung immer noch die wesentliche Rolle; mit der hier vorgestellten Baugruppe erweitern Sie Ihren Computer um alle Standard-Serien-Schnittstellen.

Ausgeknautschtes Konzept

Auf der seriellen Interface-Karte ist an Schnittstellen und Interface-ICs alles zusammengetragen, was Rang und Namen hat (**Bild 144**); busseitig erfolgt der Anschluß über eine 64polige VG-Leiste mit der gewählten Standard-ECB-Belegung.

- **Drei 16-Bit-Timer**
  Programmierbare Impulsformen und Verzögerungszeiten von 0,5 µs bis über vier Jahre.

- **Serieller Interface-Baustein**
  Programmierbare Wortlänge und Übertragungsrate, Paritätsprüfung und separate Pufferung von Sende- und Empfangsteil.

- **Treiber-ICs für V.24-Anschluß**
  ±12-V-Pegelumsetzer für sämtliche Anschlußleitungen.

- **Stromtreiber-Schnittstelle**
  Anschluß über 20-mA-Stromtreiber (*Current Loop*).

- **Separater Drucker-Anschluß**
  Serielle Schnittstelle für Standard-Drucker, programmierbare Übertragungsrate, Paritätsbit-Erzeugung.

- **Schnelles Kassetten-Interface**
  Prinzip der Phasen-Kodierung mit möglicher Übertragungsrate bis zu 4800 Bit/s.

- **Relais-Port**
  Zwei unabhängige potentialfreie Schalter für Bandgerät oder Wählen von Telefonnummern.

- **Akustischer Signalgeber**
  Eigene Portadresse für „Bell".

Gezielte Auswahl

Wie jede Stelle in einem Mikrocomputer-System, so hat auch jede der oben angesprochenen Schnittstellen ihre eigene, spezifische Adresse, über die sie von der Zentraleinheit erreichbar ist. Die gewählten Adressen, die ausschließlich Portadressen sind, fügen sich in den Interface-Adreßrahmen ein, zu dem auch das auf Seite 52 vorgestellte parallele Interface gehört (Bauanleitung „Heiße Drähte zum Computer"; **Tabelle 11**). Die Auswahllogik auf der Karte dekodiert die Adressen gemäß der Zuordnung in Tabelle 1 (**Bild 145**). Die Grundlagen zu derartigen Adreßverteilungen und -Selektierungen finden Sie im Beitrag „Das Anzapfen von Mikrocomputern" auf Seite 50. Zentraler Bestandteil dieses seriellen Interfaces ist der asynchrone Interface-Baustein 6850 von Motorola, dessen Handhabung ausführlich im Grundlagenartikel „Einzeln aus- und einsteigen" beschrieben ist (Seite 66). Seine Aus-

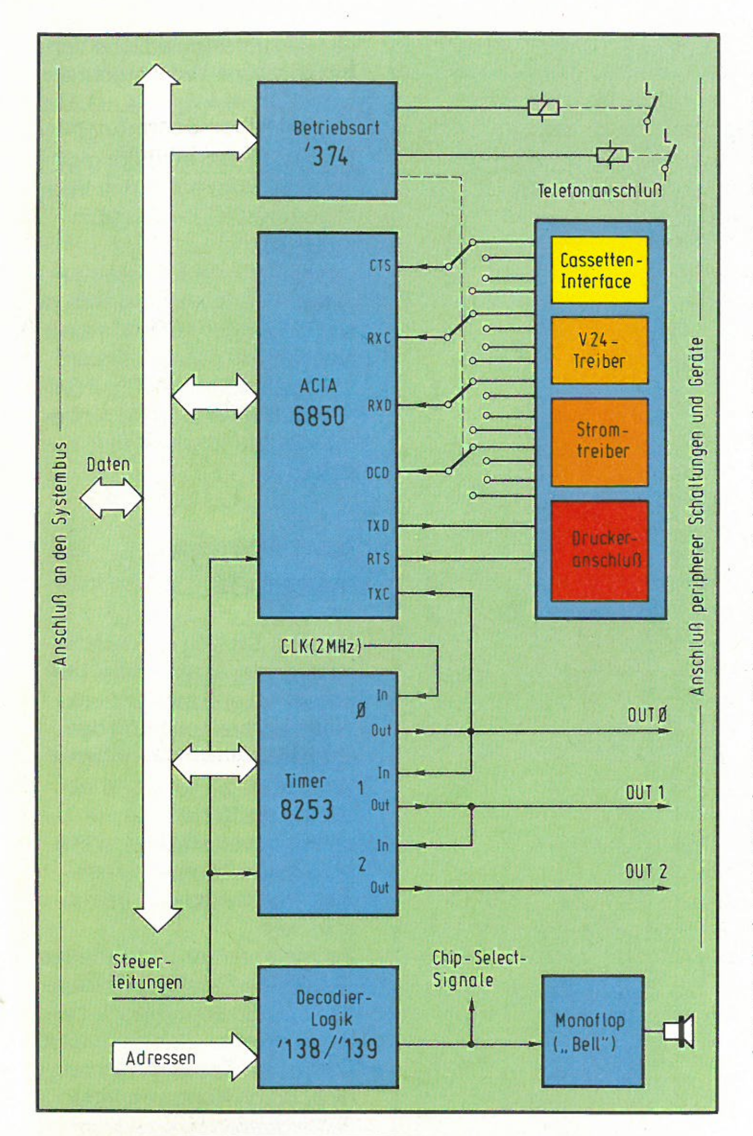


Tabelle 11. Periphere Portadressen

Adresse	Richtung	Funktion	Baugruppe
08	IN/OUT	Daten (8)	Prommer
09	OUT	Adressen (lower)	
0A	OUT	Adressen (upper)	
0B	OUT	Control	
18	IN/OUT	Daten (4)	Echtzeit-Uhr
19	OUT	Adresse/Control	
1A	OUT	Einzelschritt-Flipflop	Einzelschritt-Modul
1B		reserviert	
28/29	IN/	Adressen/Status	Video-Interface
2A/2B	OUT	Normal-/Grafik-Mode	
38	OUT	Daten (8)	Thermodrucker
39	IN/OUT	Control	
3A/3B		reserviert	
48... ...5B		reserviert	
68... ...7B		frei verfügbar	
88/**A8**		Port A	Parallel-Interface
98/**B8**		Port B	
C8/E8		Port C	
D8/F8		Control	
89	IN/OUT	Zähler 0	Timer
99	IN/OUT	Zähler 1	
C9	IN/OUT	Zähler 2	
D9	OUT	Control	
A9	OUT	Control	Serielles Interface
B9	OUT	Bell	
E9	IN/OUT	Status/Control	
F9	IN/OUT	Daten (8)	

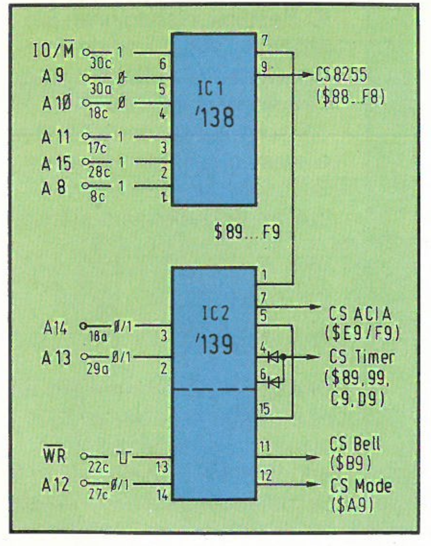


Bild 144: Das Blockschaltbild der Karte zeigt die vielfältigen Schnittstellen zur Ansteuerung sämtlicher Standard-Peripheriegeräte.

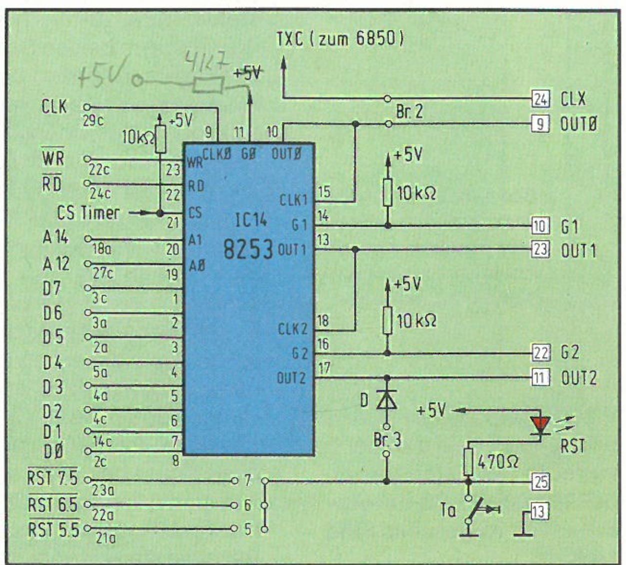
Bild 145: Zwei ICs dienen dazu, aus den ankommenden Adressen die richtigen Selektierungssignale zu erzeugen.

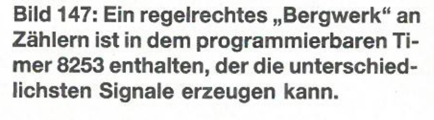
und Eingänge lassen sich über die Multiplexer IC4/IC5 an verschiedene Quellen bzw. Ziele aufschalten, wobei der Anschluß externer Peripheriegeräte in der Regel über die V.24-Pegelanpasser (IC8/IC9) erfolgt, jedoch ist darüber hinaus auch eine Stromtreiber-Schnittstelle für 20-mA-Ansteuerung vorgesehen (**Bild 146**).

Das Einstellen der Multiplexer (Eingänge A und B) geschieht über die beiden obersten Bits des Betriebsarten-Registers IC3; vier weitere Bits in diesem 8-Bit-Speicher geben die einzelnen Schnittstellen frei (Drucker, DTR-Leitung, V.24-Treiber- bzw. 20-mA-Treiber). Die untersten beiden Bits dieses Registers können zwei Relais ansteuern, die für verschiedene Anwendungen einsetzbar sind (z. B. zum Ein- und Ausschalten eines Magnetbandgerätes oder zur Ansteuerung eines Rufnummerngebers). – Im Bild oben schließlich ist ein Monoflop zur Ansteuerung eines akustischen Signalgebers erkennbar; dieses Monoflop ist ein Überbleibsel (halbes IC) vom Kassetten-Interface, und auf diese Weise bekommt es einen sinnvollen Lebensinhalt. Der Takt für den Interface-Baustein stammt vom programmierbaren Zeitgeber-IC 8253 (**Bild 147**), über dessen Behandlung Sie alle Einzelheiten im Grundlagen-Beitrag „Wohldosierte Zeitpartikel" finden (Seite 64). Am Eingang des ersten internen Zählers (Nr. 0) liegt der 2-MHz-Systemtakt, der je nach Programmierung des Timers auf die gewünschte Taktrate des 6850 heruntergeteilt wird (hierzu ist Brücke 2 zu verdrahten).

Die drei Zähler sind hardwaremäßig hintereinander geschaltet (Ausgang des einen mit dem Eingang des folgenden verbunden). Zum Sperren bzw. Freigeben der Zähler 1 und 2 sind deren GATE-Eingänge an die 25polige D-Sub-Leiste an der Frontseite zugänglich. Beim Bestücken der Brücke 3 und der RST-Leuchtdiode kann man (bei entsprechender Programmierung von Betriebsart und Teilerverhält-

Bild 146: Herzstück der seriellen Interface-Karte ist der asynchrone Interface-Baustein 6850 zur Serien-/Parallel-Wandlung und umgekehrt.




Bild 147: Ein regelrechtes „Bergwerk“ an Zählern ist in dem programmierbaren Timer 8253 enthalten, der die unterschiedlichsten Signale erzeugen kann.

nis) das unterschiedliche Verhalten dieses Timers recht anschaulich verfolgen (s. u.). Die Brücken 5, 6 und 7 ermöglichen es ferner, in Abhängigkeit von programmierten Verzögerungszeiten Interrupts auszulösen.

Zur Funktionsweise des Kassetten-Interfaces (**Bild 148**) ist an Grundlagen ebenfalls alles Notwendige gesagt („Daten vom laufenden Band“ auf Seite 68). Hinweise für die Inbetriebnahme folgen weiter unten.

Schrittweise vorwärts

Machen Sie sich vor dem Bestücken der Platine (**Bild 149**) unbedingt erst mit der Funktionsbeschreibung und den entsprechenden Grundlagen vertraut (s. o.); die Inbetriebnahme der Karte kann nur dann sinnvoll erfolgen, wenn sie Sie schrittweise vornehmen, wie hier ausführlich geschildert!

Als erste werden in die Platine die beiden 24poligen Fassungen und die 64polige VG-Leiste eingelötet. *Kein anderes IC kommt auf Fassung, um hier nicht durch Kontaktmängel Fehlerquellen zu erzeugen!* Die IC-Bestückung beginnen Sie bitte mit IC1/IC2 und den beiden *Germanium*-Dioden (IC2, Pins 4 und 6, vgl. Bild 145); danach folgt der erste Test bei eingesteckter Platine, indem Sie folgendes Endlos-Programm eingeben und starten:

```
2800 3E 01    TEST MVI  A,01
2802 D3 XX         OUT  XX
2804 07            RLC
2805 C3 02 28      JMP  TEST
```

Das Beispielprogramm gibt an eine Portadresse XX ein Byte aus, in dem immer nur ein Bit auf HIGH liegt; im nächsten Schleifendurchlauf wird dieses HIGH-Bit um eine Position verschoben (RLC) und kann so mit dem Oszilloskop oder Logikprüfstift verfolgt werden. Für „XX“ setzen Sie dabei nacheinander sämtliche Portadressen ein, die auf dieser Karte verwendet werden (vgl. Tabelle 11 und Bild 143); prüfen Sie an den jeweiligen Aus-

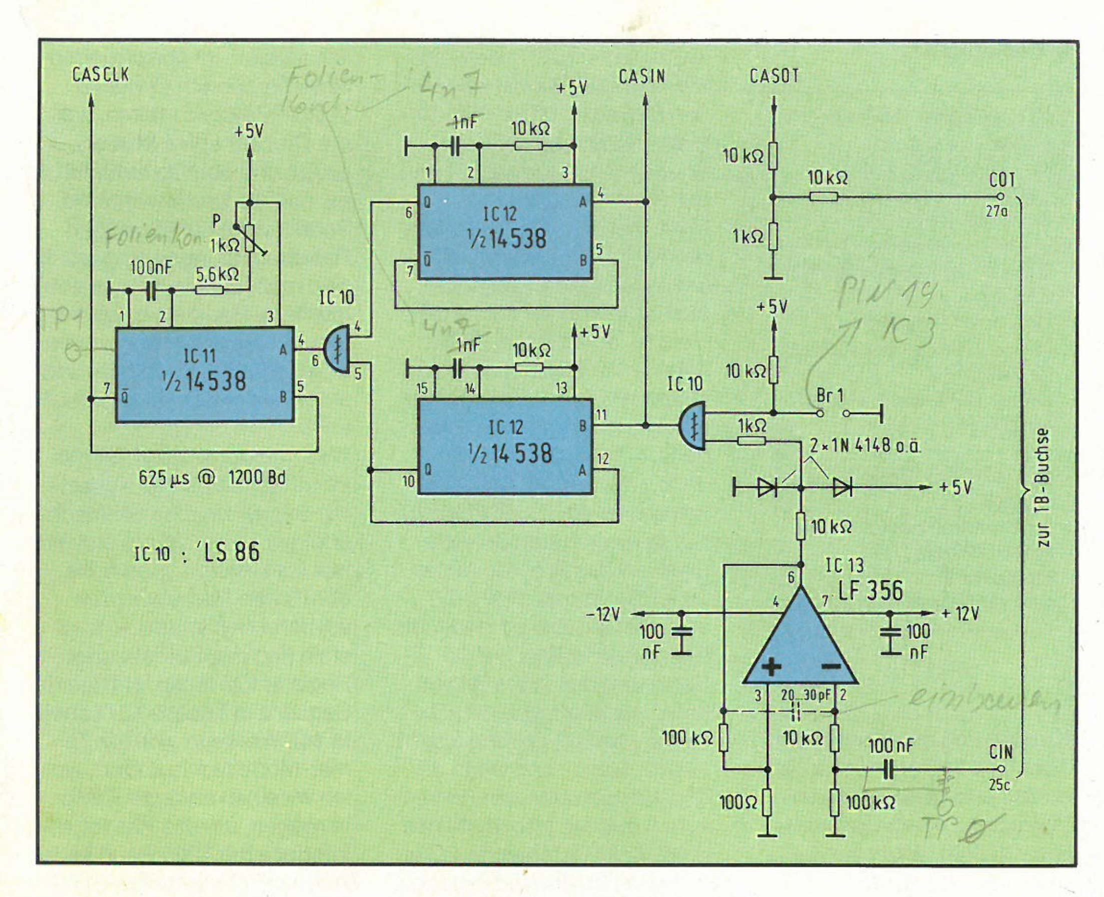


Bild 148: Das nach dem Prinzip der Phasen-Kodierung arbeitende Kassetten-Interface ermöglicht schnelle Aufnahmen bei hoher Störsicherheit.

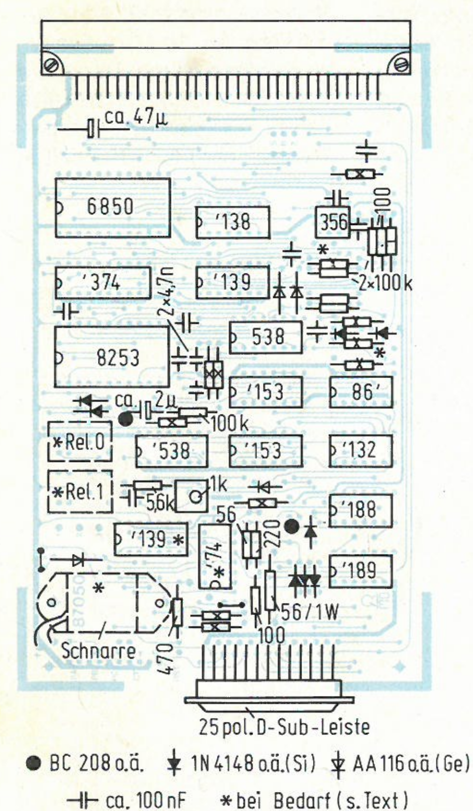


Bild 149: Bestükkungsplan für die Europa-Karte; die mit einem Stern versehenen Teile gehören nicht zum Bausatz-Lieferumfang.

gängen von IC2, ob bei der eingesetzten Adresse am passenden Ausgang (und nur an dem!) LOW-Impulse auftreten, und wenn dies der Fall ist, löten Sie auch noch IC3 ein und verfolgen an dessen Q-Ausgängen das wandernde HIGH-Bit (natürlich nur, wenn für „XX" die Adresse A9 dran ist).

Nun kommt der Timer an die Reihe; setzen Sie das IC 8253 (richtig herum!) ein und löten Sie die drei Widerstände entsprechend Bild 147 ein; zur einfachen optischen Kontrolle empfiehlt es sich ferner, auch Brücke 3 mit Ge-Diode sowie die RST-LED mit Vorwiderstand zu ergänzen, um die korrekte Arbeitsweise unseres Zeitgebers auch ohne Oszilloskop oder Frequenzzähler verfolgen zu können; die Brükken 5, 6 und 7 bleiben hierbei offen, während Brücke 2 bereits eingesetzt werden kann, da sie ohnehin gleich gebraucht wird.

Lassen Sie uns die beiden ersten Zähler so einstellen, daß jeder von ihnen durch 1024 teilt und an OUT1 die eingangs eingespeiste Taktfrequenz von 2,097 MHz (=2^{21} Hz) auf 0,5 Hz heruntergeteilt ist. Gemäß dem Beispiel in Tabelle 12 arbeitet Zähler 0 dabei binär im Mode 2, wobei nur die obere Hälfte des 16-Bit-Zählers geladen werden soll (die untere geht dabei automatisch auf „00"; Steuerwort „24" ins Control-Register mit der Zieladresse D9 überschreiben). Danach muß der Teilerfaktor für Zähler 0 geladen werden, und der beträgt gerade 0400HEX (=1024dez); hiervon braucht in der gewählten Betriebsart nur die obere Hälfte „04" überschrieben zu werden (Zieladresse 89).

Bits in langer Reihe

Zur Veranschaulichung lassen wir Zähler 1 ebenfalls durch 1024 teilen, aber im BCD-Mode-3 (Rechteckgenerator, erst unteres, dann oberes Byte laden; Steuerwort „77" ins Control-Register mit der Zieladresse D9 überschreiben); daß dieses zweite Steuerwort, das ebenfalls nach D9 transportiert wird, für Zähler Nr. 1 gilt, geht aus den beiden obersten internen Bits hervor (vgl. „Wohldosierte Zeitpartikel"). Dann folgt das Überschreiben der unteren Hälfte des Teilerfaktors 1024 („24" nach 99), gefolgt von der oberen Hälfte dieses Wertes („10" wieder nach 99). Das Sortieren und Zusammensetzen des im BCD-Format angegebenen Teilerfaktors 1024 übernimmt das IC selbständig, weil es das im Steuerwort für Zähler 1 so gesagt bekommen hat. Zum Schluß kommt die Betriebsart für Zähler 2 an die Reihe (im Beispiel Mode 3, binäres Zählen mit Laden der unteren 16-Bit-Hälfte; Steuerwort „96" ins Control-Register); nach der Vorgabe eines Teilerfaktors („0A" = 10dez. für Zähler 2 mit der Zieladresse C9) beginnt dann die insgesamt 48 Bit lange Zählerkette ihre Teilerei. Der an CLK2 ankommende Takt von 0,5 Hz wird im letzten Zähler durch Zehn geteilt, was Sie an der blinkenden LED verfolgen können (die genau je 2,5 s lang an und aus ist). Solange Sie diese Verhältnisse nicht durch Umprogrammieren eines der beteiligten Zähler ändern, blinkt Ihre Leuchtdiode (mit gewissen Einschränkungen) bis in alle Ewigkeit (also

Tabelle 12. Demonstration der Timer-Betriebsarten

Adresse	Code	Label	Befehl	Kommentar
2800	3E 24	DEMO	MVI A,24	CTL für Zähler 0 (Mode 2, bin, 1 Byte)
2802	D3 D9		OUT D9	ins Control-Register
2804	3E 04		MVI A,04	Faktor 1024 (oberes Byte = 2)
2806	D3 89		OUT 89	für Zähler 0
2808	3E 77		MVI A,77	CTL Zähler 1 (Mode 3, BCD, 2 Bytes)
280A	D3 D9		OUT D9	ins Control-Register
280C	3E 24		MVI A,24	Faktor 1024, untere Hälfte
280E	D3 99		OUT 99	für Zähler 1
2810	3E 10		MVI A,10	Faktor 1024, obere Hälfte
2812	D3 99		OUT 99	für Zähler 1
2814	3E 96		MVI A,96	CTL Zähler 2 (Mode 3, bin, 1 Byte)
2816	D3 D9		OUT D9	ins Control-Register
2818	3E 0A		MVI A,0A	Faktor 10 (dez.)
281A	D3 C9		OUT C9	für Zähler 2
281C	C3 00 00		JMP 0000	zum Monitor-Anfang zurück

auch, wenn danach irgendwelche anderen Programme ausgeführt werden). Es ist Ihnen klar, welch Eldorado sich Ihnen hier auftut; denn (über die Freigabe an GATE) können Sie eine Stoppuhr aufbauen, Frequenzen und Periodendauern messen oder auch Impulse erzeugen, die durchaus ein Tastverhältnis von Eins zu hundert Billionen (!) und mehr haben...

Wenden wir uns dem asynchronen Interface-Baustein 6850 („ACIA") zu und bestükken für dessen Inbetriebnahme zunächst die ICs 4...10 (vgl. Bild 146). Hier ist bezüglich Lötbrücken und Handhabung besondere Vorsicht geboten, weil die ICs 8 und 9 mit abartig hohen (bzw. niedrigen) Spannungen arbeiten, die bei Fehlleitung ein Himmelfahrtskommando für ihre Kollegen veranstalten! Den Test für diesen zentralen ACIA wollen wir mit einem Standard-Drucker vornehmen, der über eine serielle Schnittstelle mit 4800 Baud angesteuert wird; der Einfachheit halber verzichten wir dabei auf eine Prüfbit-Erzeugung und wählen als Wortlänge 8 Datenbits mit je einem Start- und Stopbit (Steuerwort „14" ins ACIA-Control-Register mit der Zieladresse E9).

Im Zusammenhang mit seiner Initialisierung sollte der 6850 generell softwaremäßig zurückgesetzt werden („03" ins Control-Register schreiben; **Tabelle 13**). Der Drucker-Anschluß erfolgt dabei an der 25poligen D-Sub-Leiste der Bus-Platine, wobei der Ausgang PRTOT an den R×D-Eingang des Druckers führt und dessen Rückmelde-Leitung RTS an PRTIN angeschlossen wird (gemeinsame Masseleitung nicht vergessen!).

Ein Wald voll Verwaltungsbits

Vor dem eigentlichen ACIA-Betrieb müssen noch zwei zusätzliche Dinge durchgeführt werden: Erstens muß der Timer, von dem ja der Takt für den 6850 stammt, passend programmiert werden, und zweitens ist das Register IC3 so zu laden, daß die Multiplexer IC4/5 richtig stehen und das Drucker-Freigabe-Bit „MOD5" auf HIGH liegt. Beim Timer genügt die Programmierung von Zähler 0, der im Mode 3 arbeiten und dabei durch 437 teilen soll (Verhältnis von 2,097 MHz zu 4,8 kHz für die gewählten 4800 Baud; Steuerwort „37" ins Timer-Control-Register mit der Zieladresse D9); danach folgen die Übertragung der unteren Hälfte „37" und der oberen Hälfte „04" des Teilerfaktors 437 (Zieladresse jeweils 89 = Zähler 0). Abschließend ist noch das Betriebsarten-Register IC3 zu laden („A0" nach Zieladresse A9; die obersten beiden Bits auf „10" für PRT-Betrieb, das Bit „MOD5" zur Drucker-Freigabe auf „1" und der Rest auf „0").

Nach diesen drei vorbereitenden Maßnahmen kann die (vergleichsweise spärliche) Ausgabe-Routine aufgerufen werden (ab Adresse 2838 in Tabelle 13). Die lädt das Registerpaar H,L mit der Anfangsadresse des auszudruckenden Datenblocks und liest dann zweimal das 6850-Statusregister ein (Quelladresse E9). In einer ersten Schleife maskiert das Programm das vorletzte Statusbit (ANI 02) und wartet dabei so lange, bis das Senderegister leer ist. Dann wird das vierte Bit von unten maskiert, das den Zustand der CTS-Leitung angibt (*Clar to Send*); erst wenn der Drucker hierüber seine Bereitschaft meldet, erfolgt die Ausgabe eines Datenwortes (Zieladresse F9). Eine Endabfrage ist im Programmbeispiel nicht eingebaut, aber so etwas ist für Sie als alten Software-Hasen sicherlich keine Tat mehr (schlimmstenfalls drückt man zum Abbruch RESET). Natürlich müssen im spezifizierten Speicherbereich sinnvolle (ASCII-)Zeichen stehen, weil der Drucker sonst Mist erzeugt; und ebenso natürlich ist es, daß die Drucker-interne Schnittstelle auf diejenigen Randbedingungen eingestellt sein muß, auf die wir unseren Interface-Baustein programmiert haben (Übertragungsrate, Wortlänge, Paritätsbit und automatische Druckauslösung usf.).

Was nun übrig bleibt, ist die Inbetriebnahme des Kassetten-Interfaces, das Sie (nach erfolgter Bestückung) erst einmal initialisieren, indem Sie den Timer-Takt sowie den asynchronen Interface-Baustein und das Betriebsarten-Register IC3 laden, ähnlich der Sequenz in Tabelle 13 (**Tabelle 14**). Während sich am Timer-Mode nichts ändert, wollen wir einen anderen Faktor vorgeben, um das Kassetten-Interface mit 1200 Baud laufen zu lassen (Teilerfaktor = 1748; Verhältnis von 2,097 MHz zu 1,2 kHz). An der ACIA-Programmierung (und dem vorherigen Rücksetzen) ändert sich nichts, nur das Betriebsarten-Register muß beim Kassetten-

Tabelle 13. Ansteuerung eines Druckers über den 6850

Adresse	Code	Label	Befehl	Kommentar
2820	3E 37	PRINT	MVI A,37	CTL für Zähler 0 (Mode 3, BCD,./.437)
2822	D3 D9		OUT D9	ins Timer-Control-Register
2824	3E 37		MVI A,37	Teilerfaktor (untere Hälfte)
2826	D3 89		OUT 89	in Zähler 0 überschreiben
2828	3E 04		MVI A,04	Teilerfaktor (obere Hälfte)
282A	D3 89		OUT 89	in Zähler 0 überschreiben
282C	3E 03		MVI A,03	CTL-Wort für Rücksetzen
282E	D3 E9		OUT E9	ins 6850-Control-Register
2830	3E 14		MVI A,14	CTL für 6850 (8 Bits, 1 Stop,./.1)
2832	D3 E9		OUT E9	ins 6850-Control-Register
2834	3E A0		MVI A,A0	CTL-Wort für Betriebsarten-Register
2836	D3 A9		OUT A9	ins Register IC3
2838	21 00 80		LXI H,8000	Anfangsadresse Textbuffer
283B	DB E9	LOP1	IN E9	Statuswort einlesen
283D	E6 02		ANI 02	TD-Bit maskieren
283F	CA 3B 28		JZ LOP1	warten, bis Senderegister leer
2842	DB E9	LOP2	IN E9	Statuswort erneut einlesen
2844	E6 08		ANI 08	CTS-Bit maskieren
2846	C2 42 28		JNZ LOP2	warten, bis Drucker bereit ist
2849	7E		MOV A,m	Datenbyte aus Buffer holen
284A	D3 F9		OUT F9	und an 6850 ausgeben
284C	23		INX H	Daten-Pointer hochzählen
284D	C3 3B 28		JMP LOP1	neuer Schleifendurchlauf

Tabelle 14. Ansteuerung des Kassetten-Interfaces

Adresse	Code	Label	Befehl	Kommentar
2850	3E 37	INIT	MVI A,37	CTL für Zähler 0 (Mode 3, BCD,./.1748)
2852	D3 D9		OUT D9	ins Timer-Control-Register
2854	3E 48		MVI A,48	Teilerfaktor (untere Hälfte)
2856	D3 89		OUT 89	in Zähler 0 überschreiben
2858	3E 17		MVI A,17	Teilerfaktor (obere Hälfte)
285A	D3 89		OUT 89	in Zähler 0 überschreiben
285C	3E 03		MVI A,03	CTL-Wort für Rücksetzen
285E	D3 E9		OUT E9	ins 6850-Control-Register
2860	3E 14		MVI A,14	CTL für 6850 (8 Bits, 1 Stop,./.1)
2862	D3 E9		OUT E9	ins 6850-Control-Register
2864	3E C0		MVI A,C0	CTL-Wort für Betriebsarten-Register
2866	D3 A9		OUT A9	ins Register IC3
2868	76		HLT	Stop (nur für Inbetriebnahme)

Ausgabe-Zyklus für ein Byte:

Adresse	Code	Label	Befehl	Kommentar
2869	21 00 80		LXI H,8000	Anfangsadresse Textbuffer
286C	DB E9	LOPOT	IN E9	Statuswort einlesen
286E	E6 02		ANI 02	TD-Bit maskieren
2870	CA 6C 28		JZ LOPOT	warten, bis Senderegister leer
2873	7E		MOV A,m	Datenbyte aus Buffer holen
2874	D3 F9		OUT F9	und an 6850 ausgeben
2876	23		INX H	Daten-Pointer hochzählen
2877	C3 6C 28		JMP LOPOT	neuer Schleifendurchlauf

Einlese-Zyklus für ein Byte:

Adresse	Code	Label	Befehl	Kommentar
2869	21 00 80		LXI H,8000	Anfangsadresse Textbuffer
286C	DB E9	LOPIN	IN E9	Statuswort einlesen
286E	E6 01		ANI 01	RD-Bit maskieren
2870	CA 6C 28		JZ LOPIN	warten, bis Empfangsregister voll
2873	DB F9		JN F9	Datenbyte vom 6850 einlesen
2875	77		MOV m,A	und im Textbuffer ablegen
2876	23		INX H	Daten-Pointer hochzählen
2877	C3 6C 28		JMP LOPIN	neuer Schleifendurchlauf

Betrieb mit „C0" geladen werden, um die beiden Multiplexer IC4/5 richtig einzustellen. Diese Initialisierungs-Routine muß vor jeder Aufnahme und vor jeder Wiedergabe einmal durchlaufen werden; für den gleich folgenden Abgleich (und nur während dieser Phase) ist danach ein HALT-Befehl (=76HEX) einzufügen.

Aufgepäppelter Pegel

Haben Sie diese Routine durchlaufen, erscheint an CASOT (vgl. Bild 148) kontinuierlich die voreingestellte Frequenz von 1200 Hz, da der 6850 ausgangsseitig noch ruhig bleibt. Nehmen Sie von diesem Dauerton ein paar Meter auf Band auf (Anschluß über die Tonbandbuchse der Bus-Platine) und spielen Sie das Ganze danach wieder ein. Am Anschluß CASIN (Ausgang 8 des EXOR-Gatters IC10) muß dabei das auf TTL-Pegel begrenzte 1200-Hz-Rechtecksignal sauber und ohne Mukken erkennbar sein. Sollte der Operationsverstärker hierbei zum Schwingen neigen, können Sie ihn durch ein kleines, ca. 20 pF großes, „Beruhigungs-C" daran hindern (im Bestückungsplan durch ein „*" gekennzeichnet). An den Q-Ausgängen der beiden nachgeschalteten Flipflops (IC12) müssen bei jedem Pegelwechsel des 1,2-kHz-Rechtecks kurze Pulse erscheinen, und zwar an Pin 6 bei jeder positiven und an Pin 10 bei jeder negativen Flanke am Eingang. Ist dies der Fall, müssen Sie nur noch zusehen, daß das nachfolgende Monoflop in IC11 ausgangsseitig LOW-Impulse von genau 625 µs Dauer erzeugt; das kann (und darf) nur bei jedem *zweiten* Trigger-Impuls an dessen A-Eingang passieren (Abgleich möglichst mit Oszilloskop).

Damit ist Ihr Kassetten-Interface betriebsbereit, und Sie müssen nun bei der Aufnahme nur noch berücksichtigen, daß Sie vor der Ausgabe des eigentlichen Datenblocks einen Dauerton (ohne Starten der Ausgabe-Routine) sowie ein „blindes" Führungsbyte (z. B. „FF") ausgeben müssen, bei dem der Takt bei der Wiedergabe einrasten kann (vgl. Grundlagen-Beitrag „Daten vom laufenden Band" auf Seite 68). Wiedergabeseitig bleibt noch die Frage zu klären, ob Sie die Brück Br. 1 bestücken müssen oder nicht; nach dem Einlesen eines zuvor überspielten Datenblocks ergibt sich die Antwort auf diese Frage von allein: Liegen die Daten richtig vor, brauchen Sie an Br. 1 nichts zu tun; sind aber nach dem Einlesen alle Daten invers vorhanden (also statt „3F" ein „C0"), dann müssen Sie Br. 1 einsetzen, und Ihre Kassetten-Welt ist damit in Ordnung.

Die beiden kurzen Schleifen für die Ausgabe bzw. Eingabe auf bzw. von Band enthalten wiederum keine Endabfrage, weil es hier nur um das Prinzip geht; natürlich könnte man (z. B. im Registerpaar D,E) einen Blocklängen-Zähler bei jedem Byte erniedrigen, um eine automatische Endabfrage zu erreichen. Eine weitere Erhöhung des Komforts besteht darin, bei der Ein- und Ausgabe eine Prüfsumme aller Bytes zu bilden, die nach dem Wiedereinlesen mit dem ursprünglichen Wert verglichen wird (weitere Prüfung auf Übertragungsfehler).

Schließlich steht noch die Erklärung aus für die mit einem Sternchen versehenen Teile auf dem Bestückungsplan, die nicht zum Umfang des angebotenen Bausatzes gehören. Das sind die beiden Subminiatur-Relais, die sich direkt vom Ausgangspegel von IC3 ansteuern lassen sowie eine Schnarre, die (bei entsprechender Ansteuerung) Ihren Mops regelrecht „zu Wort" kommen läßt. IC15 und IC16 haben eine elitere Funktion: Sie zeigen nämlich (über die vier angeschlossenen LEDs) an, auf welchen der parallelen Ports PA, PB, PC oder Steuer-Port (beim Parallel-Interface) gerade zugegriffen wird (**Bild 150**). Wer mag, kann diese Zusatzfunktion nachrüsten, ebenso eine kleine Taste, mit der eine manuelle Auslösung von Interrupts möglich ist (sofern eine der Brücken 5...7 verdrahtet wurde und die Interrupts entsprechend maskiert worden sind; vgl. ELO-Laborbrief Nr. 1).

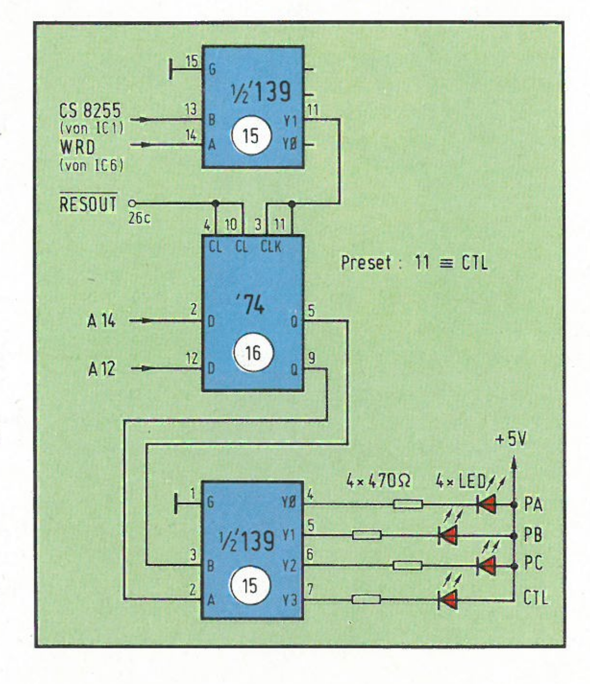


Bild 150: Auch diese Zusatzfunktion zur Anzeige des gerade aktivierten (Parallel-) Ports läßt sich noch auf der Platine unterbringen.

Im ELO-Labor-Brief Nr. 2 wird jeder Befehl anhand eines Demonstrations-Beispiels erläutert.

MOPPEL-BASIC-Befehlssatz Teil 1

Alle Eingaben können sowohl in Klein- als auch in Großbuchstaben erfolgen; Variable sind hier generell klein geschrieben.

1 Direkt-Kommandos

NEW

Den gesamten Programmspeicher löschen

Anm.: Dieser Befehl kann auch in ein Programm eingebaut werden (vgl. Abschnitt 7)

FRE(x)

Die freie Anzahl von Bytes im Arbeitsspeicher ermitteln; (x) ist ein Dummy-Argument mit beliebigem Wert

CLEAR; CLEAR x

Alle Variablen und Rücksprungadressen löschen und bei Bedarf x Bytes für den String-Speicher reservieren

Anm.: Dieser Befehl kann auch in ein Programm eingebaut werden (vgl. Abschnitt 7)

LIST; LIST x

Das im Arbeitsspeicher abgelegte Programm auf dem Bildschirm ausgeben; bei Angabe einer Zeilennummer x beginnt die Ausgabe mit dieser Zeile

LLIST; LLIST x

Das im Arbeitsspeicher abgelegte Programm auf dem Drucker ausgeben; bei Angabe einer Zeilennummer x beginnt die Ausgabe mit dieser Zeile

RUN; RUN x

Das im Arbeitsspeicher abgelegte Programm bei der niedrigsten Zeilennummer starten; bei Angabe einer Zeilennummer x beginnt die Programmausführung bei dieser Zeile

ATN(x)

Den Arcustangens von x bilden (im Bogenmaß)

SGN (x)

Das Vorzeichen von x ermitteln (Signum-Funktion; –1 bei negativem x, 1 bei positivem x, 0 bei x = 0)

ABS (x)

Den Absolutwert von x bilden (negatives Vorzeichen invertieren)

INT (x)

Den Nachkomma-Anteil der Fließkomma-Zahl x unterdrücken

3 Logische Verknüpfungen

x AND y

Das binäre Äquivalent von x und y logisch UND-verknüpfen

Anm.: Dieser Befehl ist auch im Abschnitt 6 aufgeführt

x OR y

Das binäre Äquivalent von x und y logisch ODER-verknüpfen

Anm.: Dieser Befehl ist auch im Abschnitt 6 aufgeführt

LPRINT x; LPRINT a$; LPRINT „xyz“

Die Variable x oder den String a$ oder die in Anführungszeichen stehenden Zeichen auf dem Drucker ausgeben

Anm.: Dieser Befehl ist auch im Abschnitt 1 aufgeführt

INP (x)

Daten vom Port x (x = 0...255) einlesen

Anm.: Dieser Befehl ist auch im Abschnitt 6 aufgeführt; er ist nicht zu verwechseln mit dem ähnlich lautenden 8085-Maschinenbefehl!

OUT x, y

Daten y an Port x (x, y = 0...255) ausgeben

Anm.: Dieser Befehl ist auch im Abschnitt 6 aufgeführt; er ist nicht zu verwechseln mit dem gleichlautenden 8085-Maschinenbefehl!

WAIT x, y, z

Daten vom Port x einlesen; dann mit dem binären Äquivalent von z Exklusiv-ODER-verknüpfen und das Ergebnis mit dem binären Äquivalent von y UND-verknüpfen; warten, bis das so gewonnene Ergebnis ungleich Null ist (x, y, z = 0...255)

Anm.: Dieser Befehl ist auch in den Abschnitten 6 und 7 aufgeführt; wenn die Variable z fehlt, wird dafür Null angenommen

PEEK (x)

Den Inhalt der Speicherzelle x lesen

Anm.: Dieser Befehl ist auch im Abschnitt 6 aufgeführt

PRINT

Variable oder Zeichenkette auf dem Bildschirm ausgeben

Anm.: Dieser Befehl kann auch in ein Programm eingebaut werden (vgl. Abschnitt 5)

LPRINT

Variable oder Zeichenkette auf dem Drucker ausgeben

Anm.: Dieser Befehl kann auch in ein Programm eingebaut werden (vgl. Abschnitt 5)

CLOAD

Ein Programm von der Kassette einlesen

CSAVE

Ein Programm auf Kassette ausgeben

CONT

Ein Programm nach einer Unterbrechung fortsetzen

Anm.: Dieser Befehl ist auch im Abschnitt 7 aufgeführt

BYE

Aus BASIC zurück in die Monitor-Anweisungs-Ebene gehen

2 Arithmetische Operationen

SQR(x)

Die Quadratwurzel von x bilden

EXP(x)

Die Potenz x zur Basis e (= 2,7182) bilden

LOG(x)

Den natürlichen Logarithmus von x bilden (zur Basis e = 2,7182)

SIN(x)

Den Sinus von x bilden (im Bogenmaß)

COS(x)

Den Kosinus von x bilden (im Bogenmaß)

TAN(x)

Den Tangens von x bilden (im Bogenmaß)

NOT x

Das binäre Äquivalent von x bitweise invertieren

Anm.: Dieser Befehl ist auch im Abschnitt 6 aufgeführt

4 Zeichenketten-(String-) Operationen

LEFT$ (a$, n)

Vom String a$ die n linksbündigen Zeichnen nehmen

RIGHT$ (a$, n)

Vom String a$ die n rechtsbündigen Zeichen nehmen

MID$ (a$, n, m)

Vom String a$ m Zeichen nehmen, beginnend bei Zeichen n

LEN (a$)

Die Länge des Strings a$ ermitteln

STR$ (x)

Aus der Zahl x eine Stringvariable bilden

VAL (a$)

Die numerischen Anteile der Stringvariablen a$ bilden

TI$

Die Echtzeit-Uhr auslesen; String-Format: Di, 15.03.83; 10:25:07h

5 Ein/Ausgabe-Anweisungen

INPUT x; INPUT a$

Die Variable x oder den String a$ von der Tastatur einlesen

PRINT x; PRINT a$; PRINT „xyz"

Die Variable x oder den String a$ oder die in Anführungszeichen stehenden Zeichen auf dem Bildschirm ausgeben (die PRINT-Anweisung kann durch ein „?" abgekürzt eingegeben werden)
Anm.: Dieser Befehl ist auch im Abschnitt 1 aufgeführt

POKE x,y

Die Daten y in die Speicherzelle x transportieren

Anm.: Dieser Befehl ist auch im Abschnitt 6 aufgeführt

NULL x

Bei der Ausgabe nach einem Wagenrücklauf x mal „Null" ausgeben (Anpassung an langsame Peripheriegeräte)

6 Operationen in Maschinen-Ebene

CHR$(x)

Den ASCII-Code der Zahl x bilden

ASC(„x")

Das ASCII-Zeichen x in eine Zahl umsetzen

HEX („x")

Den hexadezimalen Wert x in eine Dezimalzahl umsetzen; die HEX-Digits A...F dabei in Großbuchstaben eingeben

CALL x

Das bei Adresse x beginnende Maschinen-Unterprogramm aufrufen

Anm.: Dieser Befehl ist auch im Abschnitt 7 aufgeführt

USR x

Die bei Adresse x beginnende Anwender-Funktion aufrufen

Anm.: Dieser Befehl ist auch im Abschnitt 7 aufgeführt

INP (x)

Daten y an Port x (x, y = 0...255) ausgeben

Anm.: Dieser Befehl ist auch im Abschnitt 5 aufgeführt; er ist nicht zu verwechseln mit dem ähnlich lautenden 8085-Maschinenbefehl!

OUT x,y

Daten y an Port x (x, y = 0...255) ausgeben

Anm.: Dieser Befehl ist auch im Abschnitt 5 aufgeführt; er ist nicht zu verwechseln mit dem gleichlautenden 8085-Maschinenbefehl!



Im ELO-Labor-Brief Nr. 2 wird jeder Befehl anhand eines Demonstrations-Beispiels erläutert.

MOPPEL-BASIC-Befehlssatz Teil 2

Alle Eingaben können sowohl in Klein- als auch in Großbuchstaben erfolgen; Variable sind hier generell klein geschrieben.

6 Operationen in Maschinen-Ebene

WAIT x,y,z

Daten vom Port x einlesen; dann mit dem binären Äquivalent von z Exklusiv-ODER-verknüpfen und das Ergebnis mit dem binären Äquivalent von y UND-verknüpfen; warten, bis das so gewonnene Ergebnis ungleich Null ist (x,y,z = 0...255)

Anm.: Dieser Befehl ist auch in den Abschnitten 5 und 7 aufgeführt; wenn die Variable z fehlt, wird dafür Null angenommen

PEEK (x)

Den Inhalt der Speicherzelle x lesen

Anm.: Dieser Befehl ist auch im Abschnitt 5 aufgeführt

POKE x,y

Die Daten y in die Speicherzelle x transportieren

Anm.: Dieser Befehl ist auch im Abschnitt 5 aufgeführt

x AND y

Das binäre Äquivalent von x und y logisch UND-verknüpfen

Anm.: Dieser Befehl ist auch im Abschnitt 3 aufgeführt

x OR y

Das binäre Äquivalent von x und y logisch ODER-verknüpfen

Anm.: Dieser Befehl ist auch im Abschnitt 3 aufgeführt

USR x

Die bei Adresse x beginnende Anwender-Funktion aufrufen

Anm.: Dieser Befehl ist auch im Abschnitt 6 aufgeführt

WAIT x, y, z

Daten vom Port x einlesen; dann mit dem binären Äquivalent von z Exklusiv-ODER-verknüpfen und das Ergebnis mit dem binären Äquivalent von y UND-verknüpfen; warten, bis das so gewonnene Ergebnis ungleich Null ist (x, y, z = 0...255)

Anm.: Dieser Befehl ist auch in den Abschnitten 5 und 6 aufgeführt; wenn die Variable z fehlt, wird dafür Null angenommen

STOP

Das Programm in dieser Zeile abbrechen

CONT

Das Programm nach einer Unterbrechung fortsetzen

Anm.: Dieser Befehl ist auch im Abschnitt 1 aufgeführt

END

Das Programm abschließen; diese Anweisung ist entbehrlich

CLEAR; CLEAR x

Alle Variablen und Rücksprungadressen löschen und bei Bedarf x Bytes für den String-Speicher reservieren

Anm.: Dieser Befehl ist auch im Abschnitt 1 aufgeführt

10 übrige Anweisungen

RND (x)

Eine Zufallszahl zwischen 0 und 1 ermitteln; (x) ist ein Dummy-Argument mit beliebigem Wert

REM

Die laufende Zeile als Kommentarzeile definieren; sie bleibt bei der Programmausführung unberücksichtigt, kann dem Programm aber als Sprungziel dienen

Liste der Fehlercodes

BS	Bad Subscript Undefiniertes Feldelement (falsche Indizierung); das aufgerufene Matrix-Element liegt außerhalb der festgelegten Dimensionierung
CN	Continue Error Programmfortsetzung über CONT ist nicht möglich, weil ein Fehler vorliegt oder das Programm inzwischen modifiziert wurde

NOT x

Das binäre Äquivalent von x bitweise invertieren

Anm.: Dieser Befehl ist auch im Abschnitt 3 aufgeführt

7 Beeinflussung des Programmablaufs

GOTO x

Zur Zeilennummer x springen (unbedingter Sprung)

IF x = y GOTO z

Zur Zeilennummer z springen, wenn die Bedingung x = y erfüllt ist; andernfalls das Programm bei der nächsten Zeile fortsetzen (bedingter Sprung)

ON x GOTO a,b,c,..

Zur Zeilennummer a springen, wenn x = 1 ist, zu b springen, wenn x = 2 ist, zu c springen, wenn x = 3 ist usf. (bedingter Sprung)

FOR x=a TO z STEP n

Den Beginn einer Schleife definieren, in der die Variable x vom Anfangswert a bis zum Endwert z mit der Schrittweite n durchlaufen wird; bei fehlender Angabe der Schrittweite wird STEP=1 angenommen

NEXT x

Den Schleifenparameter x um die bei STEP angegebene Schrittweite erhöhen

GOSUB x

In das bei Zeilennummer x beginnende Unterprogramm springen (unbedingter Unterprogramm-Aufruf)

ON x GOSUB a, b, c...

In das bei a beginnende Unterprogramm springen, wenn x=1 ist, nach B springen, wenn x=2 ist, nach c springen, wenn x=3 ist, usf. (bedingter Unterprogramm-Aufruf)

RETURN

Ein Unterprogramm abschließen und ins aufrufende Hauptprogramm zurückspringen

CALL x

Das bei Adresse x beginnende Maschinen-Unterprogramm aufrufen

Anm.: Dieser Befehl ist auch im Abschnitt 6 aufgeführt

NEW

Den gesamten Programmspeicher löschen

Anm.: Dieser Befehl ist auch im Abschnitt 1 aufgeführt

8 Deklarationsanwendungen

LET

Einen Wert zuweisen; diese Anweisung ist entbehrlich

DIM x (y)

Die Matrixvariable x für maximal y Werte dimensionieren; bei mehrdimensionalen Matrizen werden die für y eingesetzten Werte durch Kommata getrennt

DEF FN a (x)=expr.

Den Funktionsnamen a als den mathematischen Ausdruck „expr." definieren; (x) ist ein Dummy-Argument mit beliebigem Wert

DATA x, y, z...

Die Daten x, y, z... (Variable oder Strings) bereitstellen; sie werden mit der READ-Anweisung verschiedenen Variablen zugewiesen

READ a; READ a$

Die im Datenfeld bereitgestellte Variable a (bzw. den String a$) fortlaufend abrufen

RESTORE

Den bei jeder READ-Anweisung um Eins erhöhten Daten-Pointer zurücksetzen (erneuter Zugriff auf dieselben Daten)

9 Cursor-Steuerung

POS (x)

Die Cursor-Position in der laufenden Zeile ermitteln (Anzahl der in der laufenden Zeile bereits ausgegebenen Zeichen ermitteln); (x) ist ein Dummy-Argument mit beliebigem Wert

SPC (x)

X Leerschritte (Blanks) erzeugen

TAB (x)

Den Cursor um x Stellen nach rechts bewegen

DD	Double Dimension Mehrfachdefinition eines Feldes; nachdem bereits eine DIM-Anweisung gegeben wurde, ist eine weitere für dieselbe Matrix entdeckt worden
FC	Function Call Error Bei einem Funktionsaufruf liegt ein Parameter außerhalb des zulässigen Bereichs
ID	Illegal Direct Diese Anweisung ist als Direkt-Kommando nicht zulässig
LS	Long String Ein String überschreitet die maximal zulässige Länge von 255 Zeichen
NF	NEXT without FOR Es fehlt die Spezifikation des Schleifenanfangs durch eine passende FOR...TO...STEP-Anweisung
OD	Out of Data Entweder wurden mit DATA zu wenig Daten bereitgestellt, oder es sind bereits alle Daten mit READ ausgelesen worden
OM	Out of Memory Der Arbeitsspeicher ist voll, weil entweder das Programm zu lang ist, zu viele Variablen benutzt wurden oder zu viele Sprungadressen aufgetreten sind (durch FOR/NEXT-Schleifen bzw. Unterprogramm-Aufrufe)
OS	Out of String Space Der verfügbare String-Speicherbereich ist durch zu viele oder zu lange Strings belegt
OV	Overflow Das Ergebnis einer Rechenoperation überschreitet den zulässigen Zahlenbereich
RG	RETURN without GOSUB Es taucht ein RETURN-Befehl auf, ohne daß vorher ein Sprung ins Unterprogramm erfolgt ist
SN	Syntax Error Es liegt eine Verletzung des Eingabeformats vor, beispielsweise eine fehlende Klammer oder ein falscher Befehl
ST	String too Complex Ein String ist zu komplex; er muß in mehrere kleine Strings aufgespalten werden
TM	Type Mismatch In einer Zuweisung kollidieren String und numerische Variable bzw. statt einer Variablen wurde ein String übergegeben oder umgekehrt
UF	Undefined Function Die aufgerufene Funktion ist zuvor nicht definiert worden
US	Undefined Statement Ein aufgerufenes Sprungziel existiert nicht im Programm
/0	Division by Zero Verbotene Division durch Null



8085-Maschinenbefehle

Tabellarische Zusammenstellung der 8085-Maschinenbefehle

DATENTRANSPORTBEFEHLE

Register r	A	B	C	D	E	H	L	m
MOV A,r	7F	78	79	7A	7B	7C	7D	7E
MOV B,r	47	40	41	42	43	44	45	46
MOV C,r	4F	48	49	4A	4B	4C	4D	4E
MOV D,r	57	50	51	52	53	54	55	56
MOV E,r	5F	58	59	5A	5B	5C	5D	5E
MOV H,r	67	60	61	62	63	64	65	66
MOV L,r	6F	68	69	6A	6B	6C	6D	6E
MOV m,r	77	70	71	72	73	74	75	--
MVI r,XX	3E XX	06 XX	0E XX	16 XX	1E XX	26 XX	2E XX	36 XX

Registerpaar	B	D	H	SP
LXI rp,XXYY	01 YY XX	11 YY XX	21 YY XX	31 YY XX

Registerpaar	B	D	H	PSW
PUSH	C5	D5	E5	F5
POP	C1	D1	E1	F1

LDA Adr	3A YY XX	STA Adr	32 YY XX
LHLD Adr	2A YY XX	SHLD Adr	22 YY XX

Registerpaar rp	B	D
LDAX rp	0A	1A
STAX rp	02	12

XTHL	E3
SPHL	F9
XCHG	EB
IN XX	DB XX
OUT XX	D3 XX

ARITHMETISCH/LOGISCHE BEFEHLE

ADI XX	ACI XX	SUI XX	SBI XX	ANI XX	XRI XX	ORI XX	CPI XX
C6 XX	CE XX	D6 XX	DE XX	E6 XX	EE XX	F6 XX	FE XX

Register r	A	B	C	D	E	H	L	m
ADD r	87	80	81	82	83	84	85	86
ADC r	8F	88	89	8A	8B	8C	8D	8E
SUB r	97	90	91	92	93	94	95	96
SBB r	9F	98	99	9A	9B	9C	9D	9E
ANA r	A7	A0	A1	A2	A3	A4	A5	A6
XRA r	AF	A8	A9	AA	AB	AC	AD	AE
ORA r	B7	B0	B1	B2	B3	B4	B5	B6
CMP r	BF	B8	B9	BA	BB	BC	BD	BE
INR r	3C	04	0C	14	1C	24	2C	34
DCR r	3D	05	0D	15	1D	25	2D	35

Registerpaar	B	D	H	SP
INX rp	03	13	23	33
DCX rp	0B	1B	2B	3B
DAD rp	09	19	29	39

RLC	07	RRC	0F	DAA	27
RAL	17	RAR	1F	CMA	2F

SPRUNGBEFEHLE

Bedingung	unc.	C	NC	Z	NZ	P	M	PE	PO
JMP XXYY	C3 YY XX	DA YY XX	D2 YY XX	CA YY XX	C2 YY XX	F2 YY XX	FA YY XX	EA YY XX	E2 YY XX
CALL XXYY	CD YY XX	DC YY XX	D4 YY XX	CC YY XX	C4 YY XX	F4 YY XX	FC YY XX	EC YY XX	E4 YY XX
RET	C9	D8	D0	C8	C0	F0	F8	E8	E0

PCHL	E9

n	0	1	2	3	4	5	6	7
RST n	C7	CF	D7	DF	E7	EF	F7	FF
Sprungziel	ØØØØ	ØØØ8	ØØ1Ø	ØØ18	ØØ2Ø	ØØ28	ØØ3Ø	ØØ38

ÜBRIGE

NOP	00	STC	37	EI	FB	RIM	20
HLT	76	CMC	3F	DI	F3	SIM	30

Franzis'

Warum microtronic-Besitzern der Einstieg in die Mikrocomputerei so leicht fällt.

Computer sind Problemlöser. Aber nur für den, der sie auch beherrscht. Und der es versteht, ein Problem in kleinere Teilaspekte zu zerlegen, die man dem Computer häppchenweise verabreichen kann.

Das wahre Kunststück dabei ist nicht etwa, Formeln oder Anweisungen in irgendeiner Programmiersprache zu formulieren. Sondern die Hauptarbeit ist nach wie vor, überhaupt diese Formeln und Anweisungen zu finden, also das Problem exakt zu analysieren. Das setzt eigentlich sogar eine für manchen ganz neue Denkweise voraus: Problemlösungen Schritt für Schritt.

Um das zu lernen, dafür gibt es microtronic. Ein richtiger Computer mit intelligenten Befehlen, die es Ihnen leicht machen, Hardware und Software zu verstehen, ohne sich mit der Undurchschaubarkeit einer höheren Sprache zu belasten. Mit überschaubaren 4 Bit Wortbreite, aber mit einem Konzept, das den modernsten 16- und 32-Bit-Prozessoren ähnelt.

Lernen Sie verstehen, wie Computer „denken". Dabei hilft Ihnen die didaktisch aufgebaute Begleitliteratur, die mit Ihrem microtronic mitgeliefert wird.

Denn das ist das Einzigartige am microtronic: Sie erfassen schnell, wie aus Problemen Programme werden.

microtronic ist Teil des *electronic* Experimentier-Systems

Es umfaßt vom Transistor bis zum Mikrocomputer die ganze Elektronik: Preise Stand 1984. Umverbindliche Preisempfehlung

2059 Netzgerät	DM 33,90
2060 Compact-Studio	DM 59,90
2061 Ergänzungspackung für 2060	DM 95,–
2065 Radio-Technik, Opto-Elektronik	DM 139,–
2069 Ergänzungspackung für 2065	DM 55,–
2070 Studio-Center	DM 179,–
2072 IC-Verstärkertechnik	DM 48,–
2075 Digital-Technik	DM 85,–
2079 Ergänzungspackung – Steckbausteine	DM 13,50
2087 Netzstrom-Schaltgerät	DM 69,50
2089 Ergänzungspackung – IC-Fassungen	DM 14,90
2095 Cassetten-Interface	DM 139,50
5964 Schwachstrom-Spezial-Relais	DM 16,90
2090 Mikro-Computer microtronic	DM 299,–
2094 Programm-Buch Computer-Spiele für microtronic 2090	DM 19,90

Der microtronic wird für jeden ernsthaft an der Mikrocomputerei Interessierten zu einem idealen Einstieg. Und das für nur DM 299.

BEZUGS-MÖGLICHKEITEN

Beim Elektronik-Fachhandel, bei größeren Buchhandlungen oder direkt beim Franzis-Verlag, Karlstraße 37–41, 8000 München 2, Telefon (0 89) 51 17-2 39/-3 80.

Bei Bezug ab Verlag können Sie unter drei Möglichkeiten wählen, wobei den genannten Verkaufspreisen jeweils 3,70 DM Porto hinzuzurechnen sind:

1. Vorauszahlung auf unser Postscheckkonto München Nr. 813 75-809
2. Zusendung eines Schecks
3. Bestellung per Nachnahme (zuzüglich 1,70 DM Nachnahme-Gebühr)

Bitte denken Sie an genaue Bestell- und Absenderangaben.

Das electronic-Experimentier-System erhalten Sie in der Schweiz beim **Verlag Thali AG**, CH-6285 Hitzkirch und in Österreich beim **Fachbuch Center Erb**, Amerlingstraße 1, A-1061 Wien.



In Zusammenarbeit mit dem Elektronik-Magazin ELO